Commodity-based Scalable Visualization

Constantine Pavlakos, Sandia National Laboratories
Randall Frank, Lawrence Livermore National Laboratory
Allen McPherson, Los Alamos National Laboratory
Greg Humphreys, Stanford University
Matthew Eldridge, Stanford University
Adam Finkelstein, Princeton University
Alan Heirich, Compaq Computer Corporation

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Commodity-based Scalable Visualization

Introduction

Parallel Rendering Overview

Commodity-based Graphics Cluster Components and Clustering Issues

- 3D Graphics HW, Interconnects, etc.
- Advanced Displays

Overview of Existing Systems and Current Results

- DOE/ASCI-lab Efforts
- Stanford's Multi-Graphics Efforts
- Princeton's Scalable Display Efforts
- Compaq's Compositing Network

An Open-Source, Parallel Rendering API Effort

Closing Remarks

Commodity-based Scalable Visualization

Introduction

Constantine “Dino” Pavlakos
Sandia National Laboratories

SIGGRAPH 2001
EXPLORE INTERACTION AND DIGITAL IMAGES

The ASCI Program

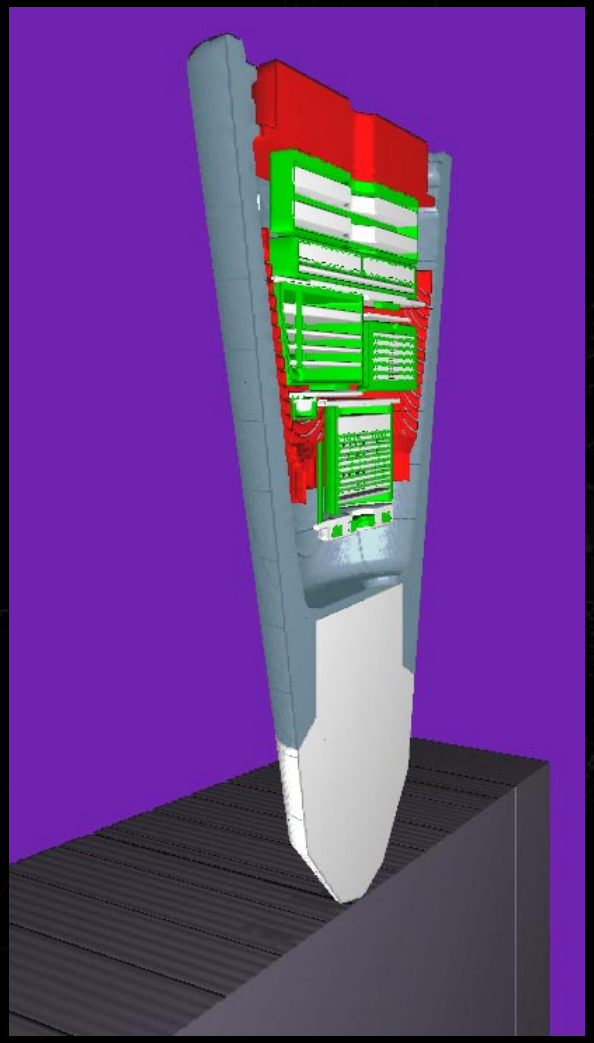
ASCI - Accelerated Strategic Computing Initiative

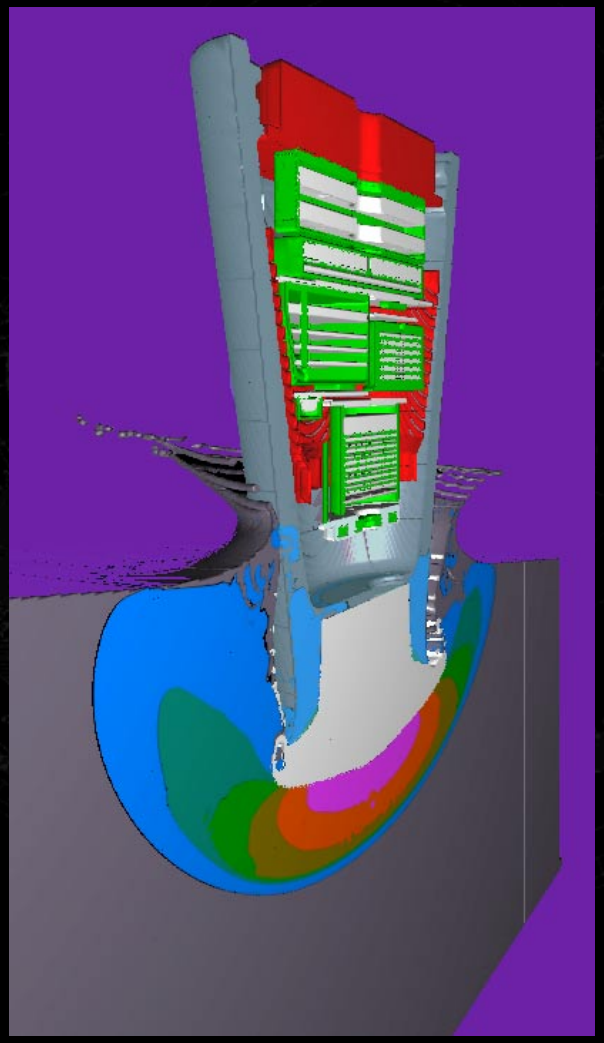
- DOE Program
- Part of nuclear Stockpile Stewardship Program
- Ensure continuing safety, performance, and reliability of nuclear stockpile without testing
- By providing unprecedented capability for simulation and modeling to support needed confidence levels

VIEWS - Visual Interactive Environment for Weapons Simulation

- ASCI sub-program element
- Provides enabling technologies and infrastructure for data management and visualization
- Enable "See and Understand"

Virtual Experiments





- Full-System Simulations
- Complex
- High-Fidelity
- Extremely large data

Shock Physics Calculation, Re-entry Body Impact

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

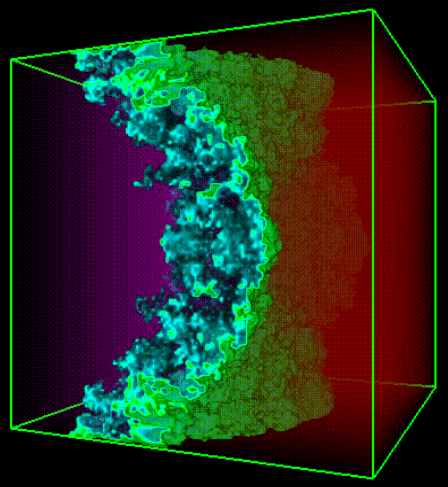
Intricate Detail

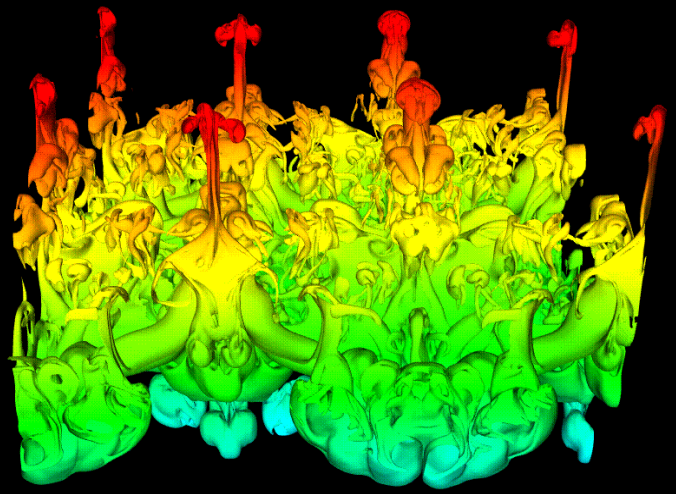
Simulation of Richtmyer-Meshkov Instability.

Two gases, which are initially separated by membrane pushed against wire mesh, are subjected to Mach 1.5 shock.

ASCI SST Machine
960 nodes.

Visualization by M. Duchaineau, J. Hobson, D. Schikore, LLNL





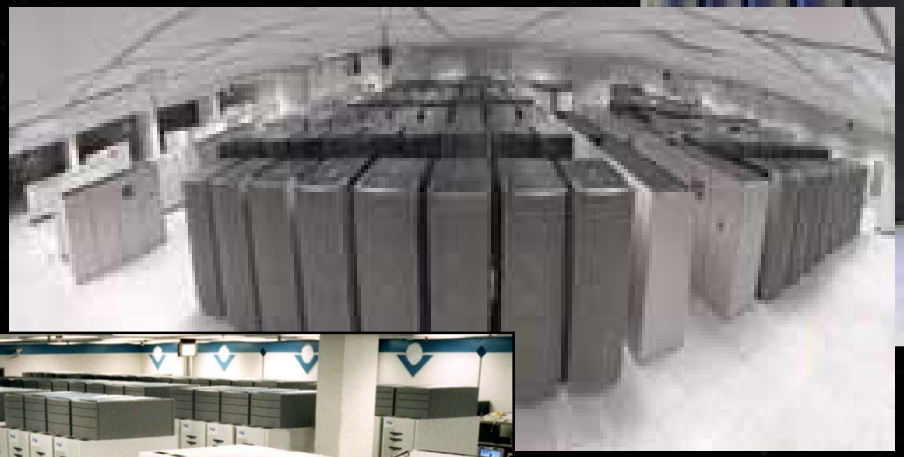
Very late time non-linear development of an unstable shocked interface between two fluids. The density isosurface shown was extracted and rendered from a 73.5 million cell AMR mesh.

Image: Bob Kares/Jamie Painter, LANL

ASCI Computation: One TeraFLOP and beyond ...

Mountain Blue

ASCI White



ASCI Red

- 3 TeraOps or more at each ASCI Lab now (thousands of processors)
- 10 TeraOps at LLNL ("White")
- 30 TeraOps machine announced for LANL
- 100 TeraOps in 2004

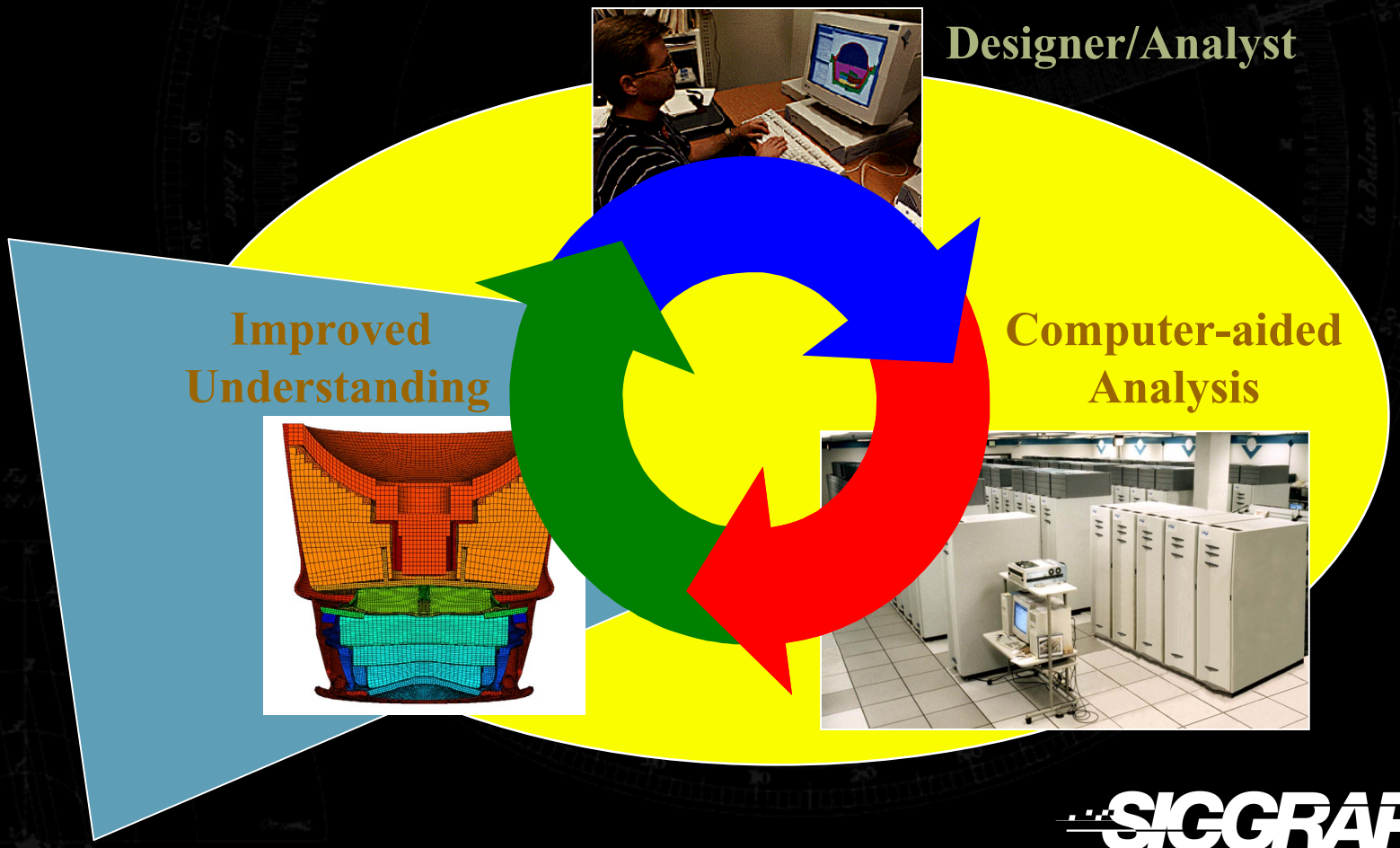
100 TFLOPS - Sizing the Problem

	Structured			Unstructured		
Compute Platform	1 TFlop	10 Tflop	100 Tflop	1 TFlop	10 Tflop	100 Tflop
Number of Cells	300 Million	1.5 Billion	7.5 Billion	100 Million	500 Million	2.5 Billion
Memory	300 GB	1.8 TB	10.5TB	160GB	800GB	6.0TB
Single Timestep	53GB	324GB	1.9TB	28GB	140GB	950GB
Compressed 100 Step Database	0.352TB	2.2TB	12.8TB	0.280TB	2.8TB	19 TB

Database Size & Transfer

Channel	Speed (Mbytes/sec)	One TFLOP Database (hours)	100 TLFOP Database (hours)
Standard Ethernet	0.5	200	11,000
ATM O/C-3	13	7.5	410
ATM O/C-12	50	2.0	110
100 Base T Ethernet	10	9.8	530
Gigabit Ethernet	100	1.0	53
1 TeraFLOP compressed database	0.350 Terabytes - 100 DVD or 540 CD		
100 TeraFLOP compressed database	19.0 Terabytes - 5,400 DVD or 29,300 CD		

Design/Analysis Cycle



ASCI/VIEWS is pushing out in Key Technology Areas

- Data Handling/Services
- User Interfaces and Visualization/Interaction Environments
- Intelligent/Hierarchical/Distributed Data Exploration
- Distance Visualization
- *Displays and Resolution*
- *Scalable Visualization/Rendering*

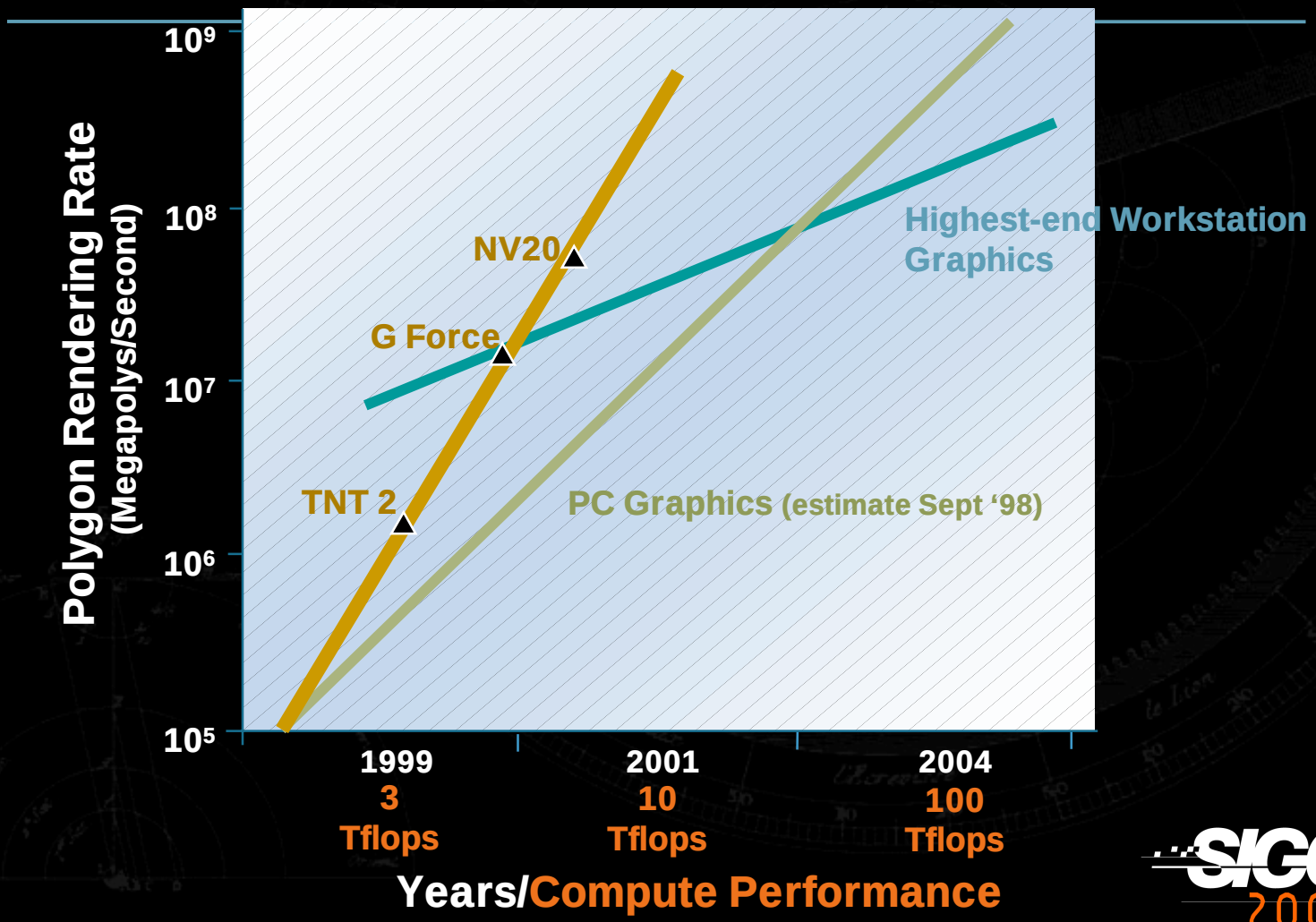
ASCI Visualization Needs

- ASCI needs about 1000 times the performance of today's high end rendering engines in 2004 (unaffordable)
- Today's high end systems are not designed to scale beyond a modest number of pipes

	Today's High-end Technology	2004 Needs
Surface Rendering	~2.5 Million polygons per second per graphics pipe	20 Billion polygons per second (aggregate)
Pixel Fill Rate	~.1 Gpixel per raster manager	200 Gpixel (aggregate)
Display Resolution	16 Mpixel (4K x 4K)	64 Mpixel (8K x 8K)

Today's performance figures based on experiences with ASCI-lab applications.

Graphics Platforms



SIGGRAPH 2001
EXPLORE INTERACTION AND DIGITAL IMAGES

Clustering PC Graphics Cards - A Promising Approach

Price / Performance

Technology - Leverages a large industry ... games!

Future - Strong, exploding market.

Practical ??? - Unknown. Many technical problems to be solved such as compositing, I/O, image quality, and efficiency to name a few.

Parallel Graphics: Scalability and Communication

Matthew Eldridge

Stanford University

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

Why Scalable Graphics?

Demanding applications

- Scientific visualization
- Photorealistic rendering
- Virtual reality
- Large-scale displays

Massive parallelism

- Billion transistor chips

Scalability is a time machine

- Build bigger machines now from commodity parts

Scalability and Communication

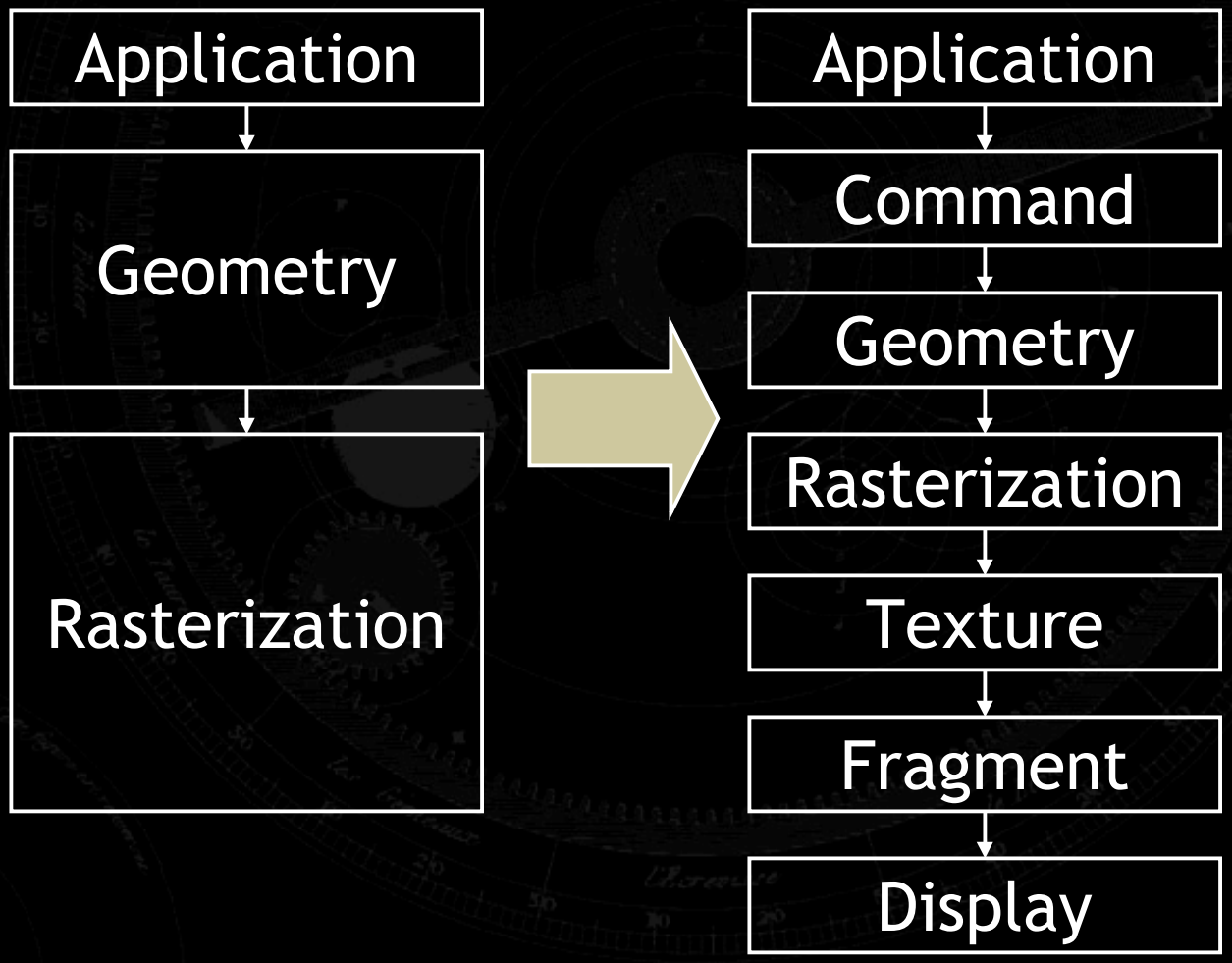
Scalability

- Design a single component (graphics pipeline) and replicate it to increase performance

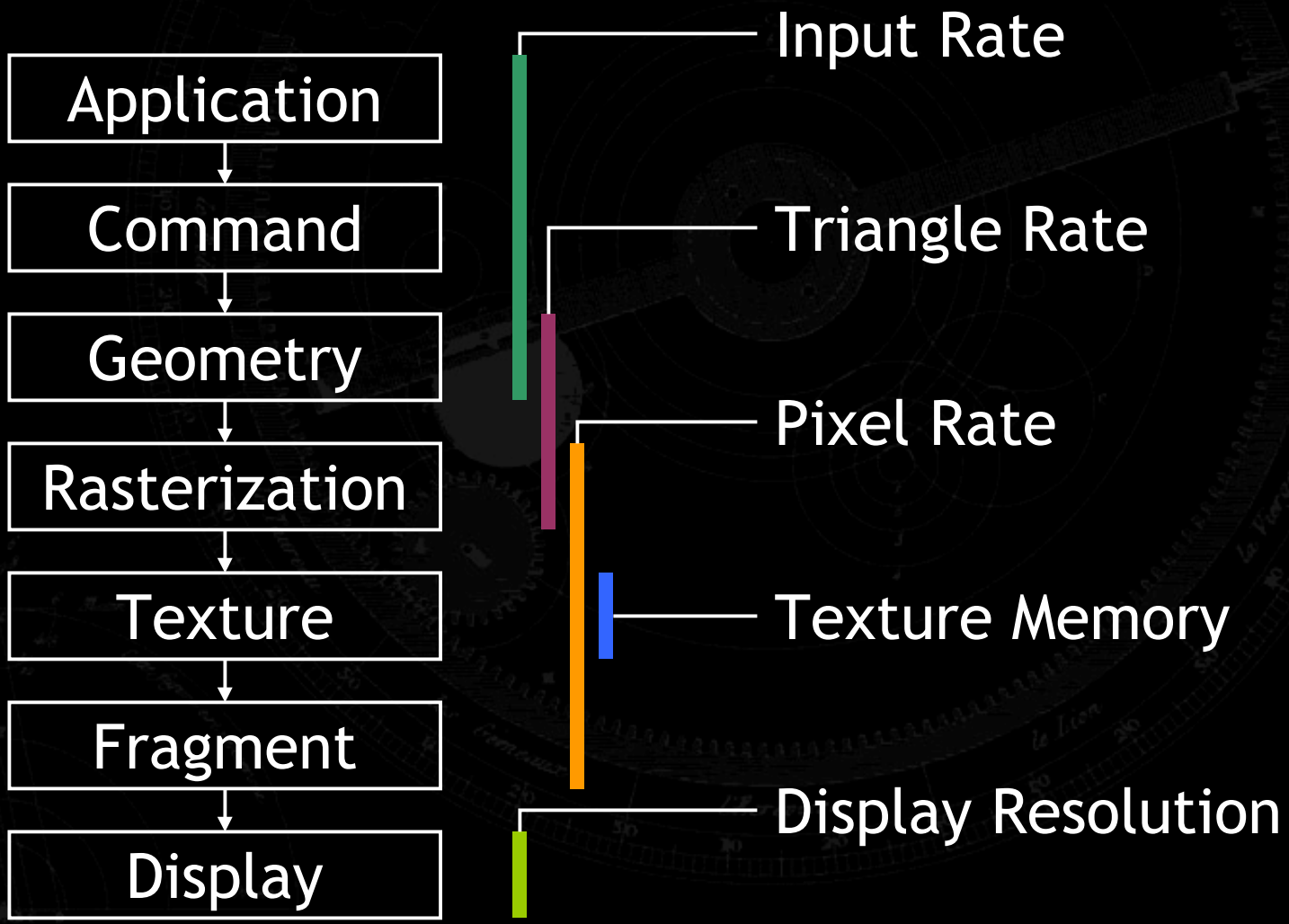
Communication

- Connects pipelines, allowing parallel work to be load balanced

Graphics Pipeline



Measuring Performance



Sources of Parallelism

Task parallelism

- Graphics pipeline

Data (primitive) parallelism

- Object(Geometry)-parallel
- Image-parallel

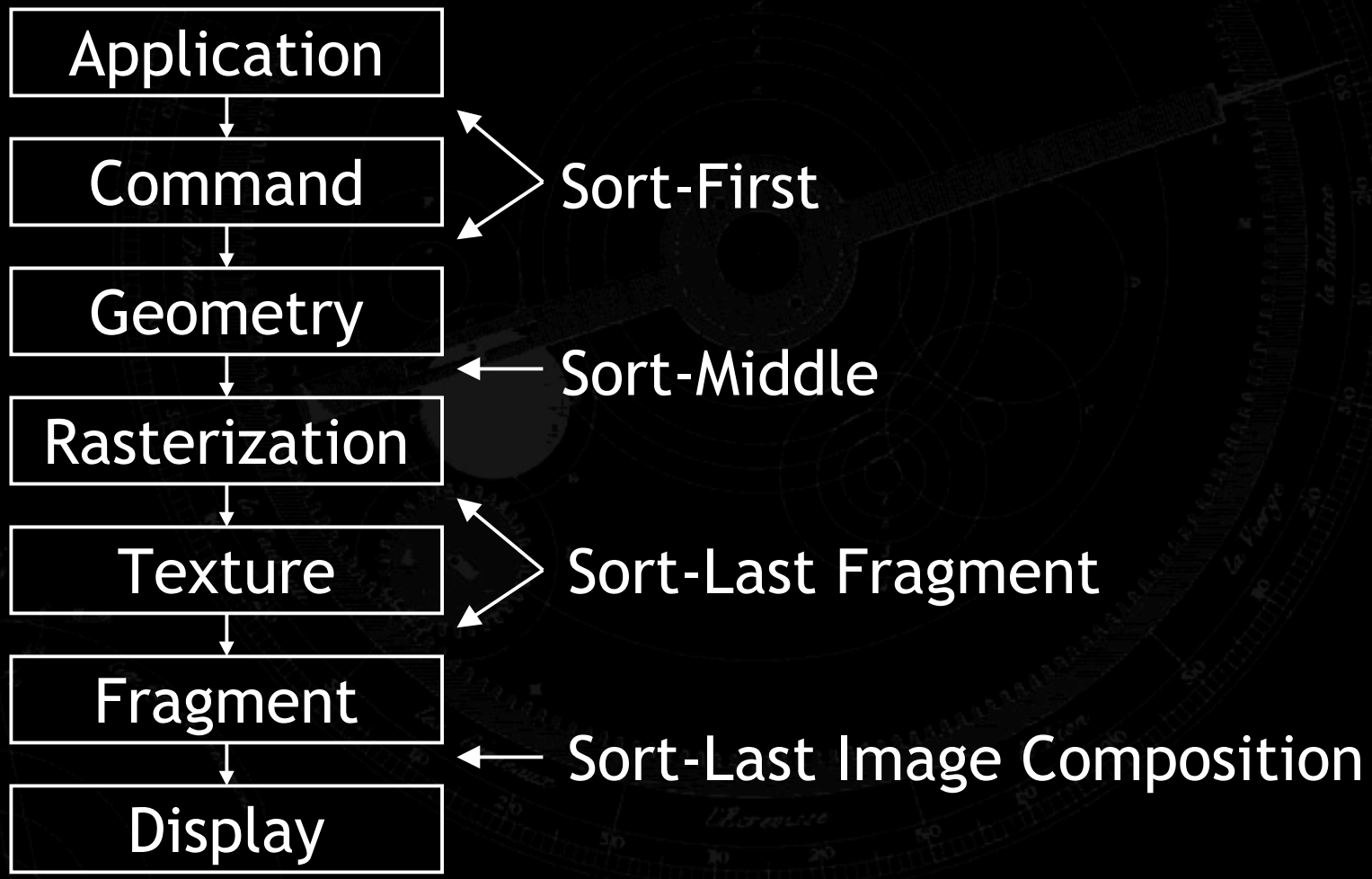
Sorting Taxonomy

A Sorting Classification of Parallel Rendering,
Molnar, Cox, Ellsworth, Fuchs

Distribution, Sorting, Routing Taxonomy

Communication in Parallel Graphics Systems,
Eldridge (to appear)

Communication (Sorting) Taxonomy



Communication Taxonomy (cont.)

Load balancing

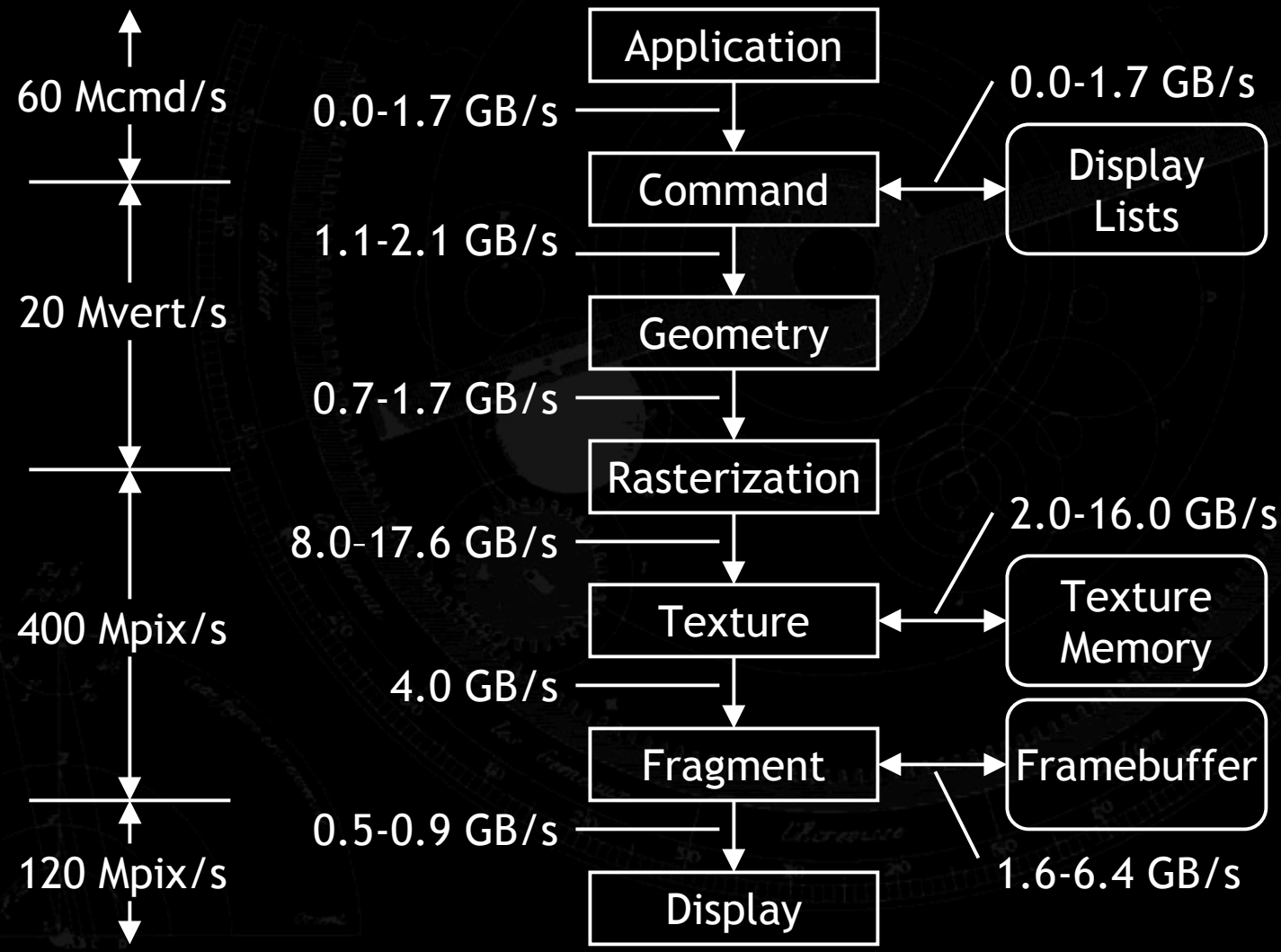
- Primitive work varies greatly
- High spatial locality over short periods

$\Rightarrow$ Balance redundant work (overlap) against load balance

Communication

- Sorting: Object $\rightarrow$ Image
- Distribution: Object $\rightarrow$ Object
- Routing: Image $\rightarrow$ Image
- Broadcast has cost proportional to parallelism

Communication Requirements



SGI InfiniteReality

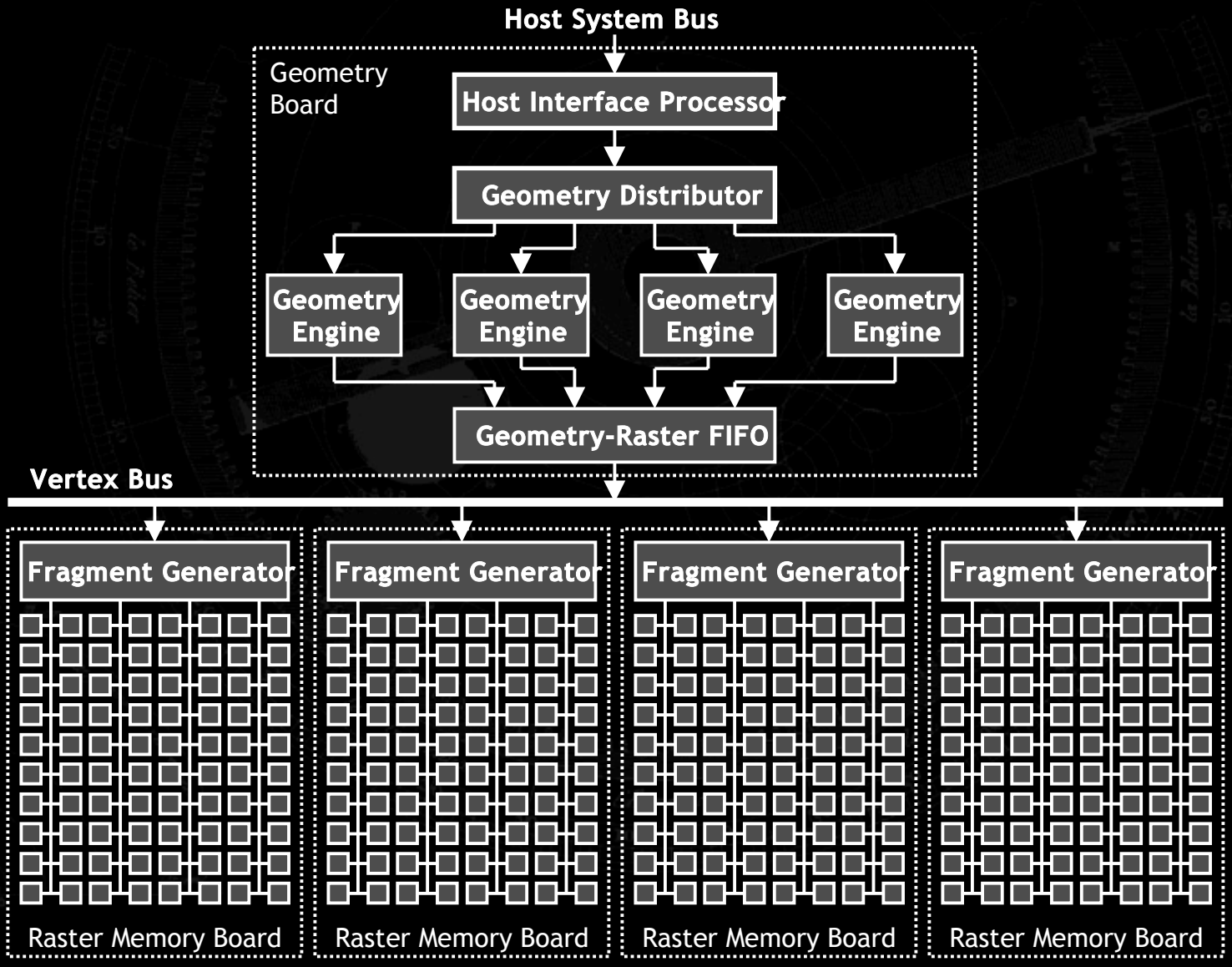
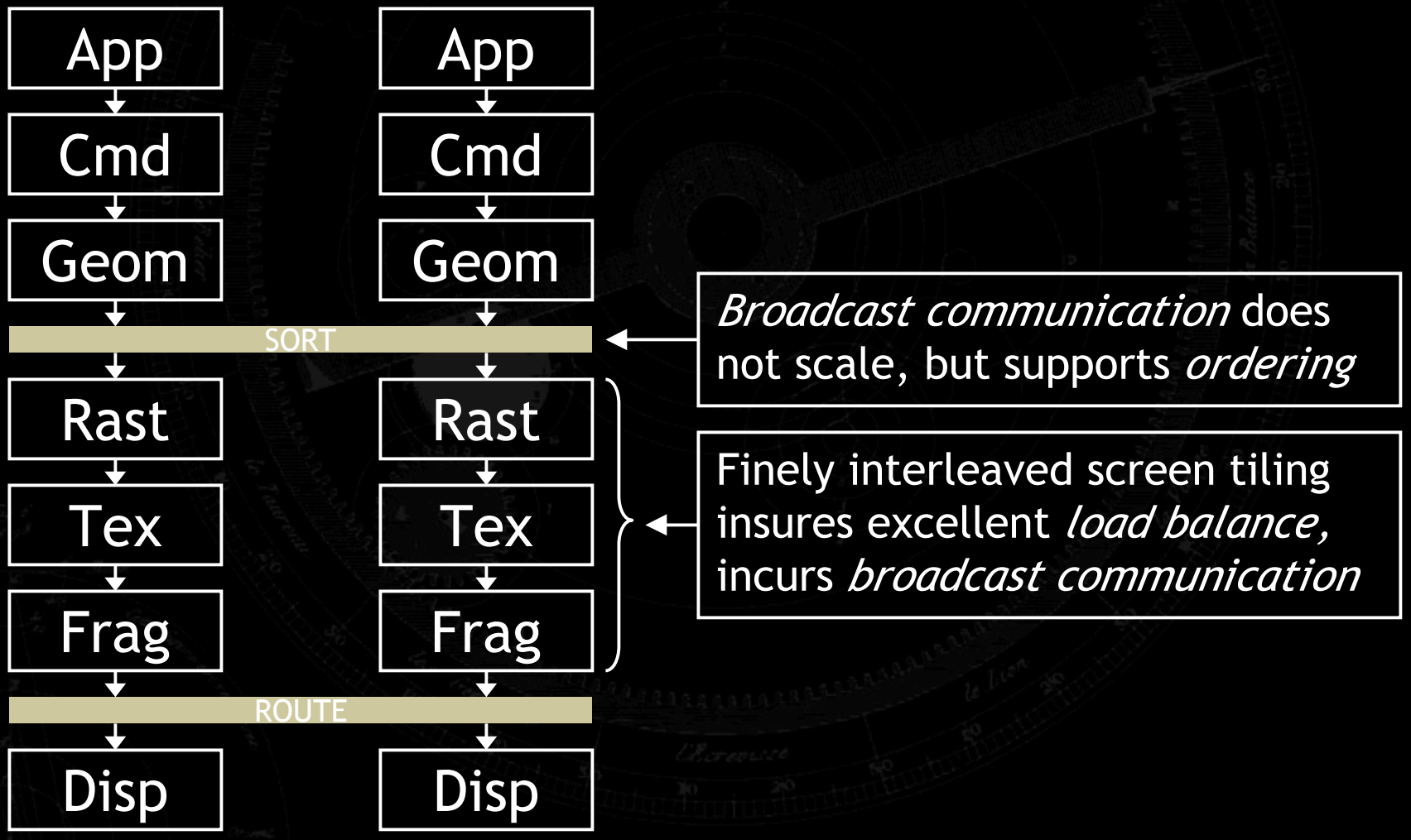


Image Parallel

Interleave → Broadcast

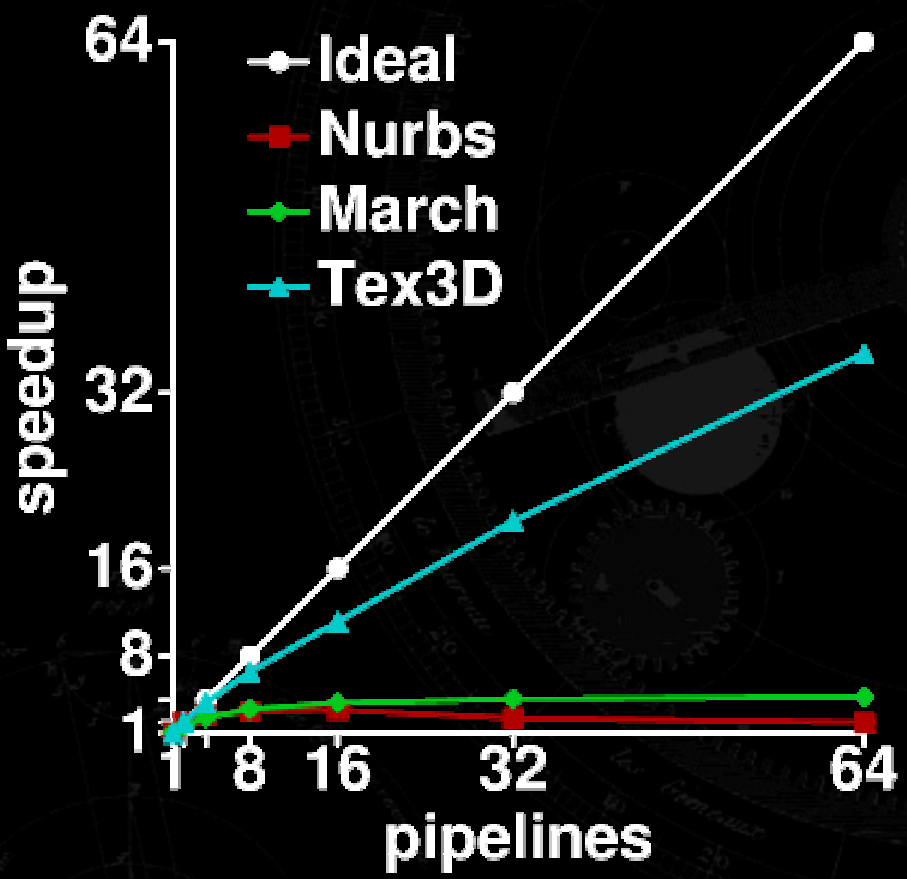
Tiles → Point-to-Point

Sort-Middle Interleaved



SGI Graphics Workstations: RealityEngine, InfiniteReality

Sort-Middle Interleaved Results



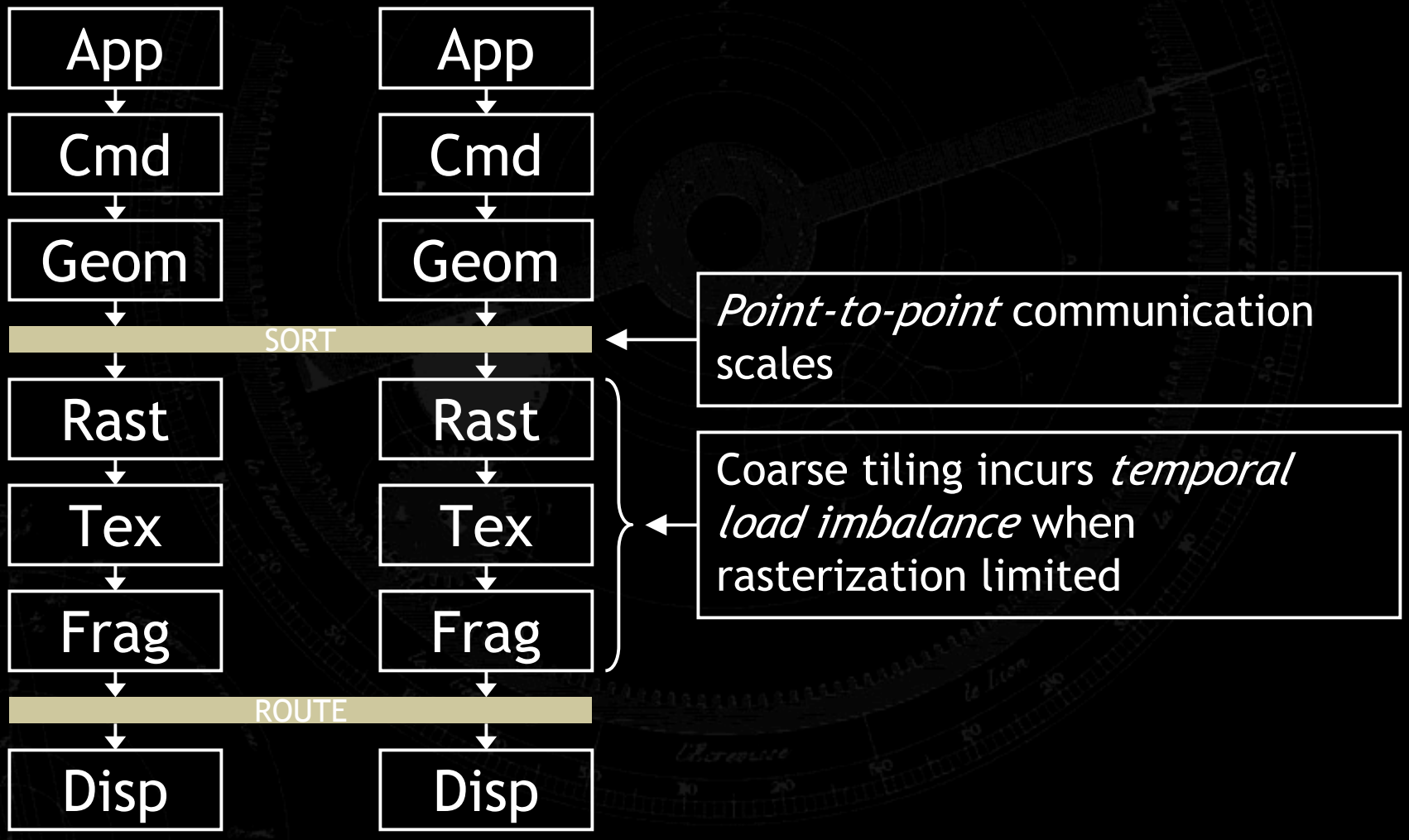
Marching cubes

- Broadcast of small primitives limits performance
- Can build to target max parallelism (4 to 8-way)

Volume rendering

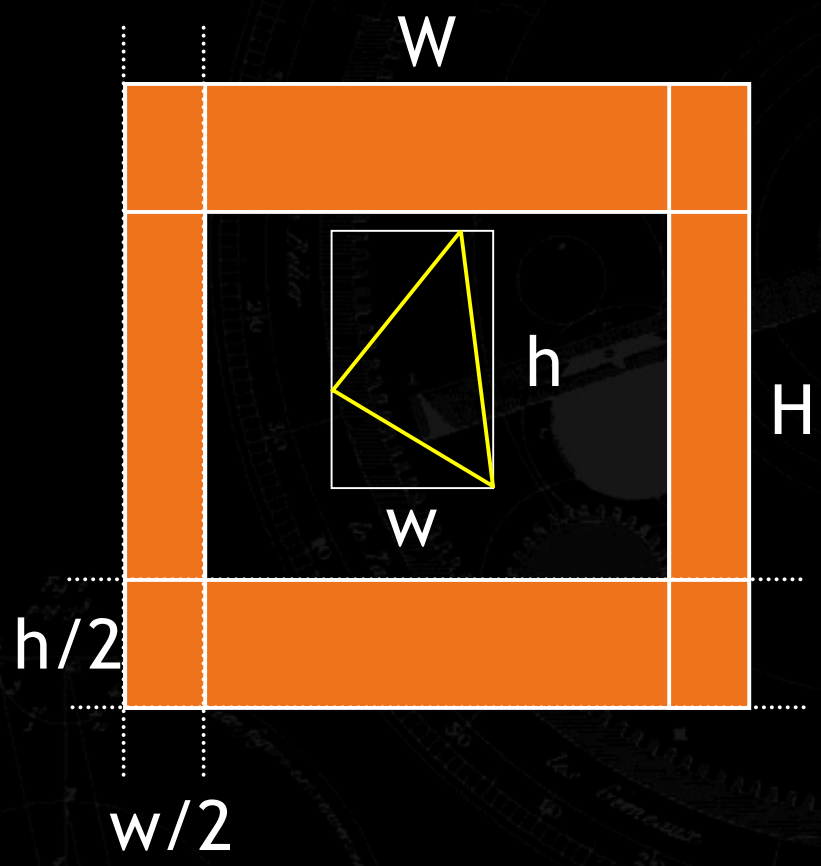
- Large primitives scale well
- Texture locality limits scalability
- Duplicated textures

Sort-Middle Tiled



UNC PixelPlanes, Stanford Argus

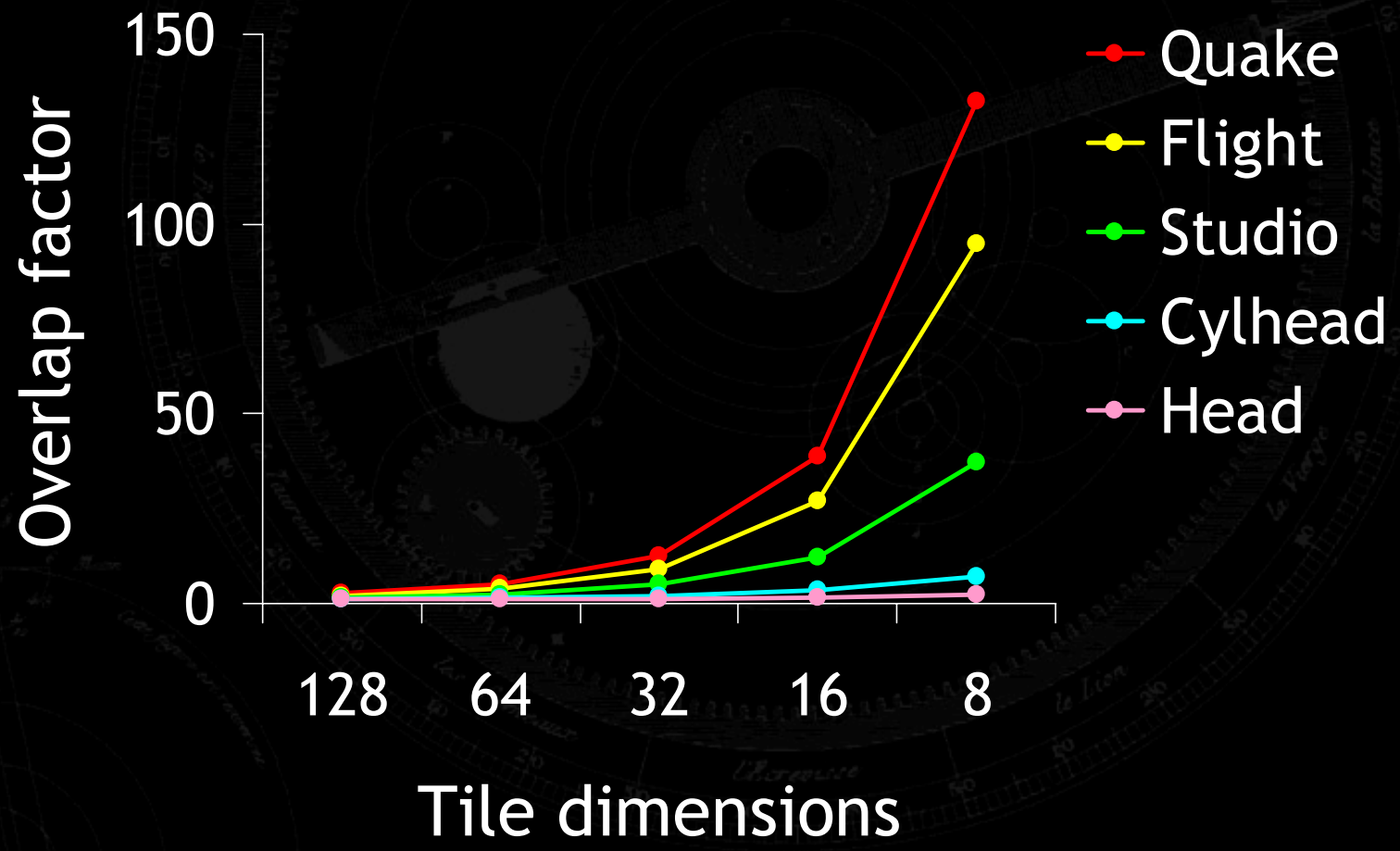
The Overlap Factor



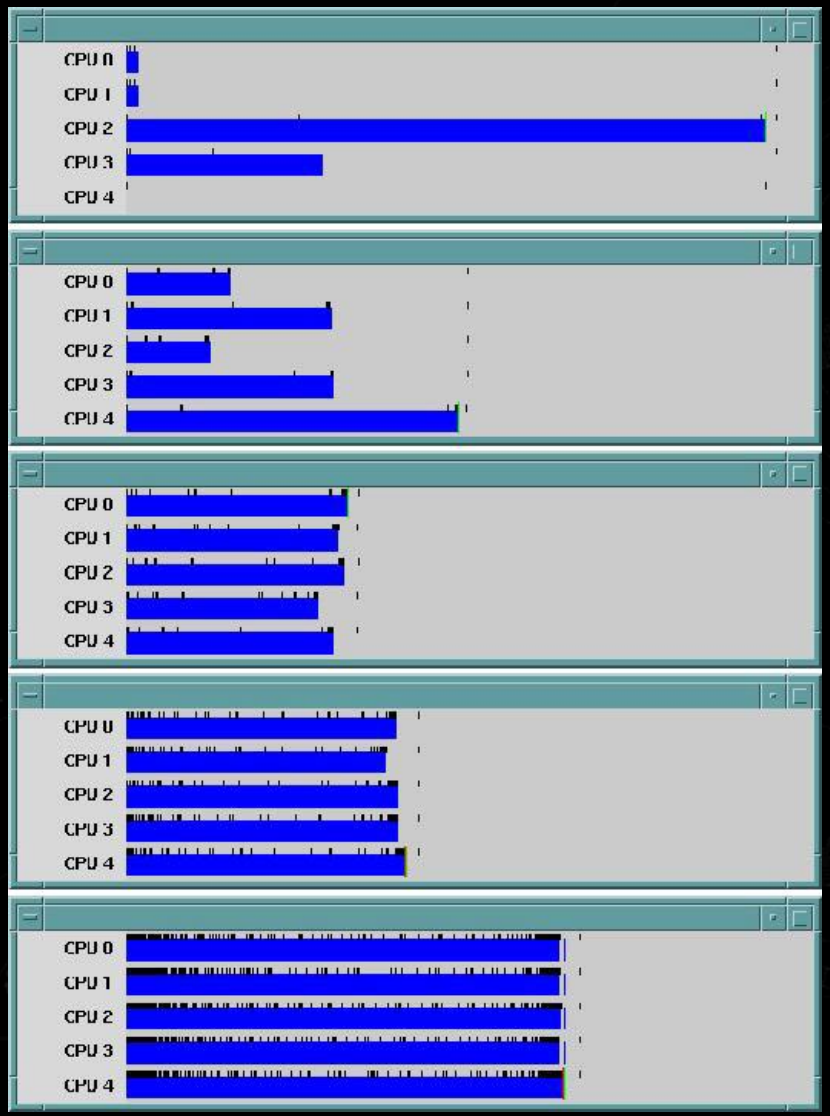
$$O = \left(\frac{H+h}{H}\right)\left(\frac{W+w}{W}\right)$$

Molnar-Eyles Formula

The Overlap Factor



Load Balancing: Rasterization



- **Large tiles: few tasks, greater variation in work**

  → bad load balance

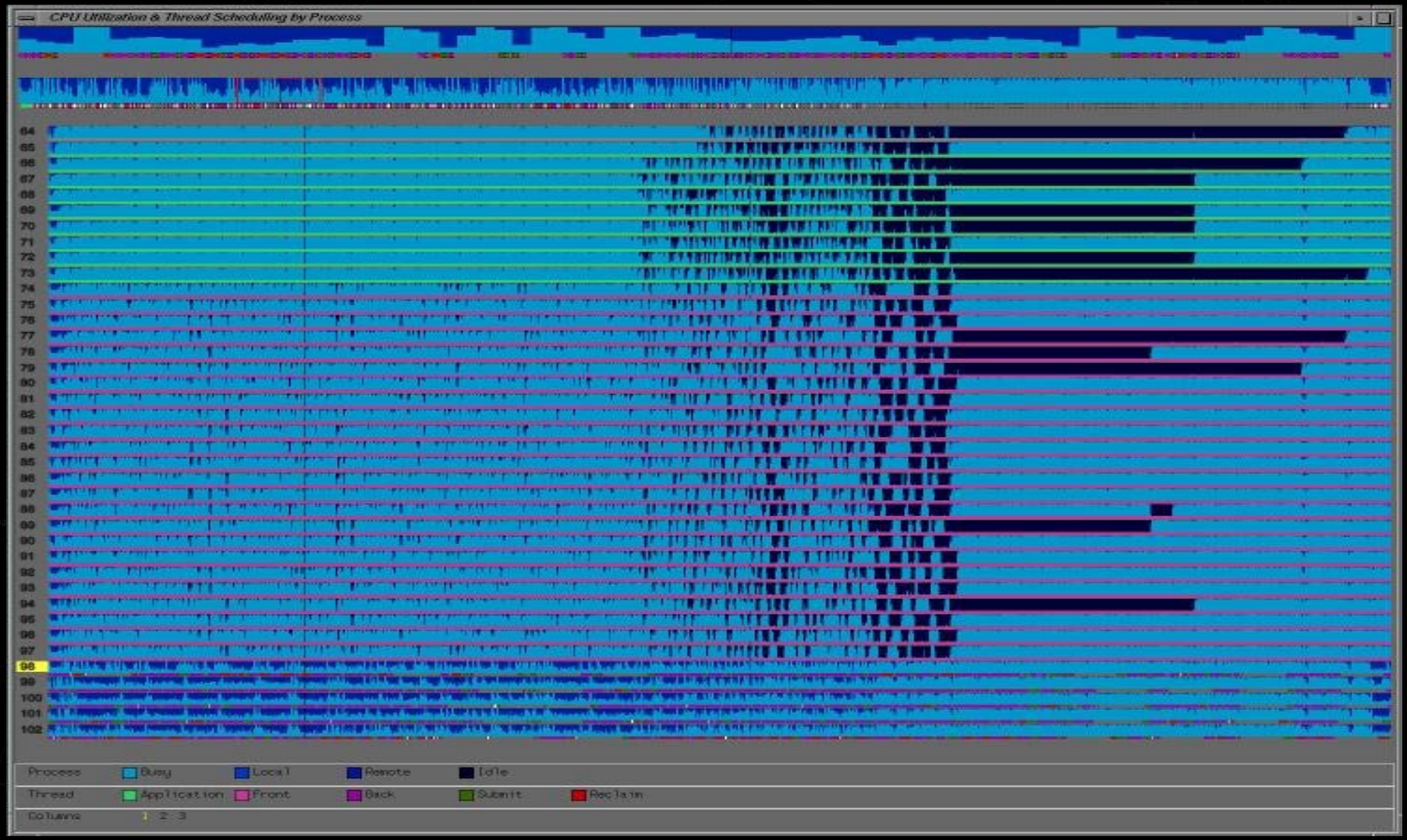
- **Medium tiles: more tasks, low overlap**

  → good load balance

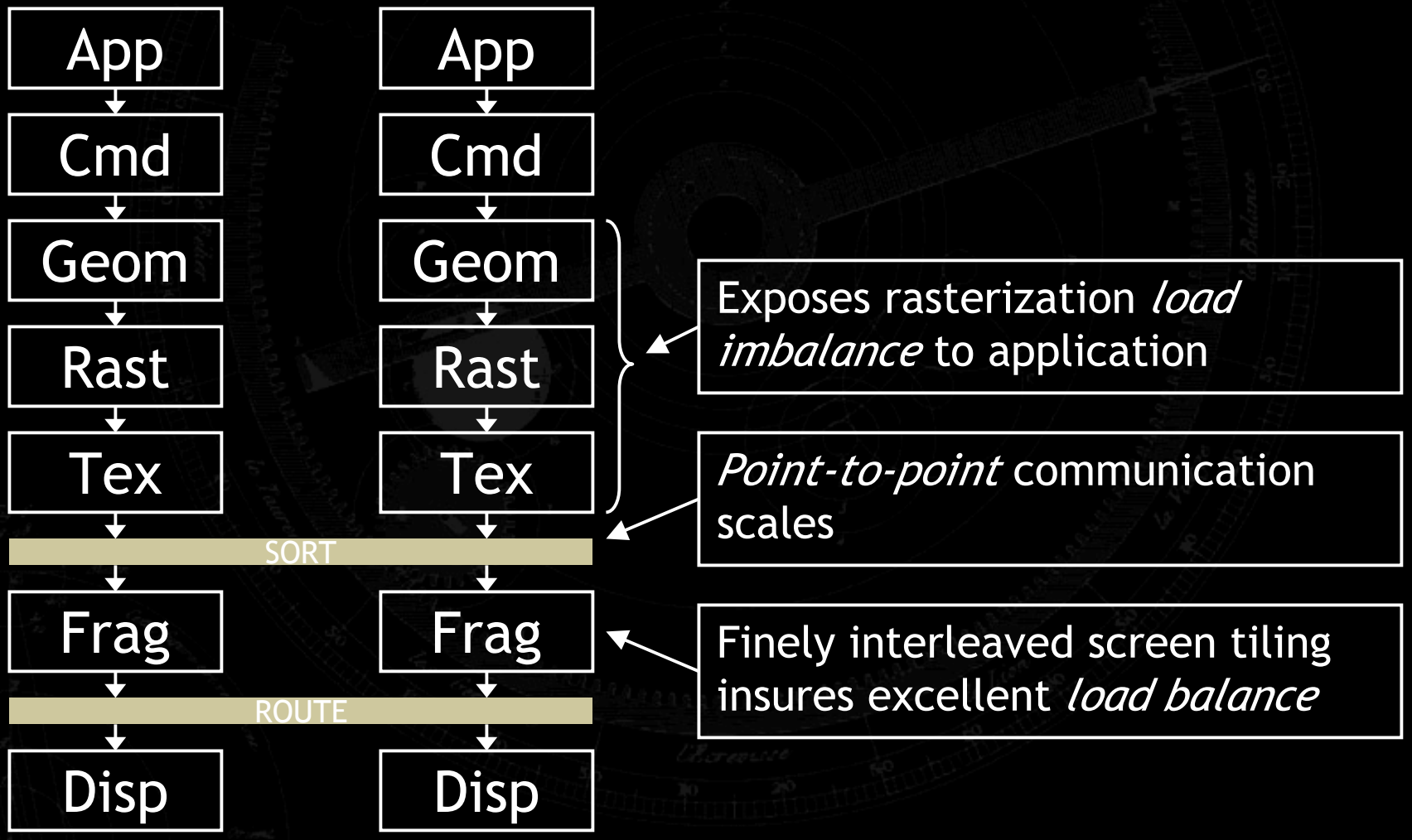
- **Small tiles: high overlap/more communication**

  → best load balance but redundant work

Temporal Load Imbalance

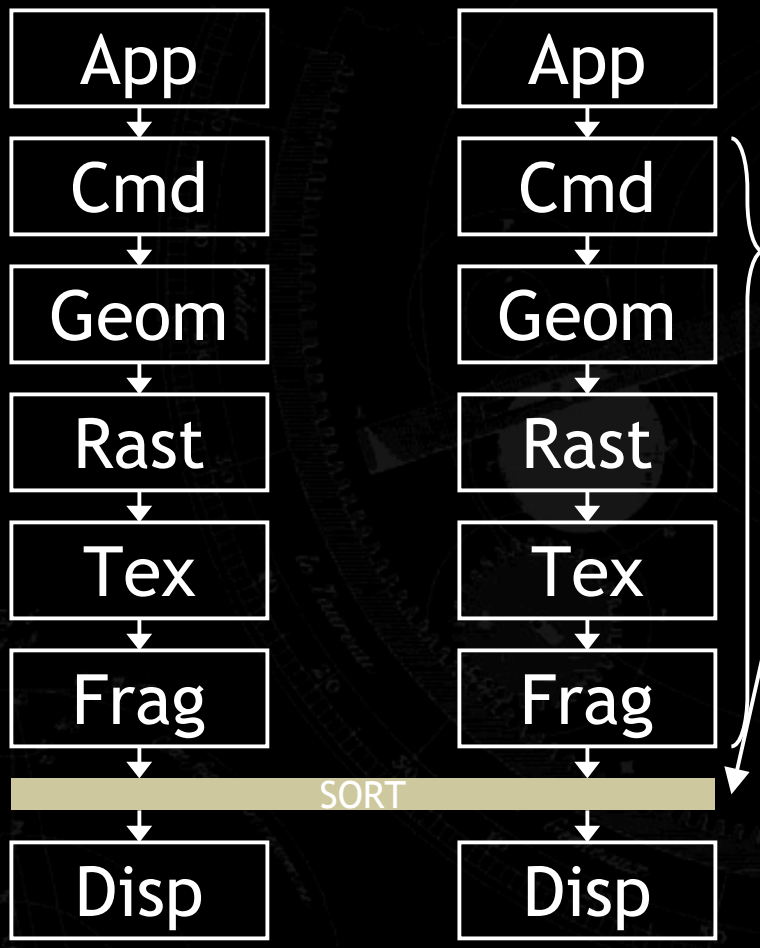


Sort-Last Fragment



Kubota Denali, E&S Freedom 3000

Sort-Last Image Composition



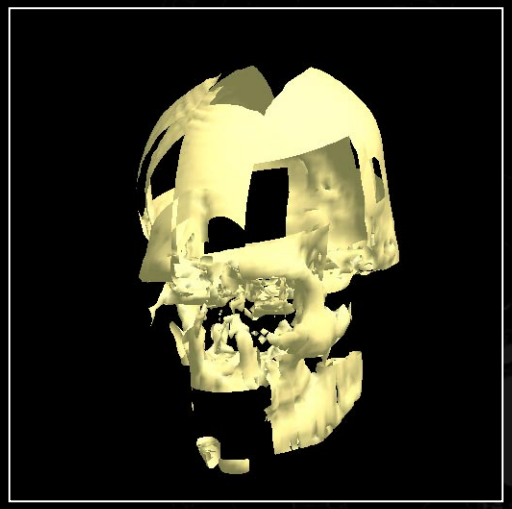
Exposes rasterization *load imbalance* to application

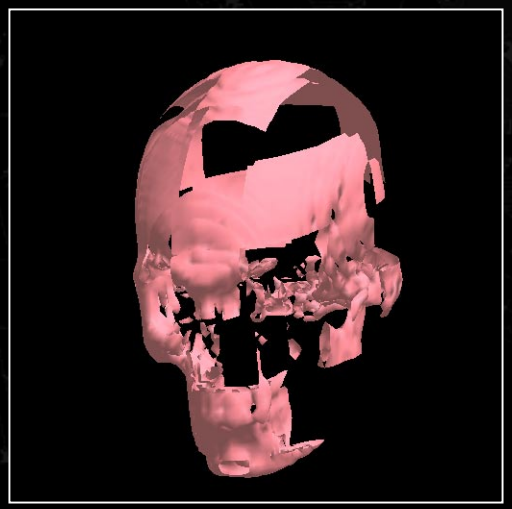
Point-to-point ring interconnect *scales*, sort after fragment processing *loses ordering*

- **Requires *more bandwidth* than SL-Fragment, may be more readily built**
- **Maps well onto clusters**

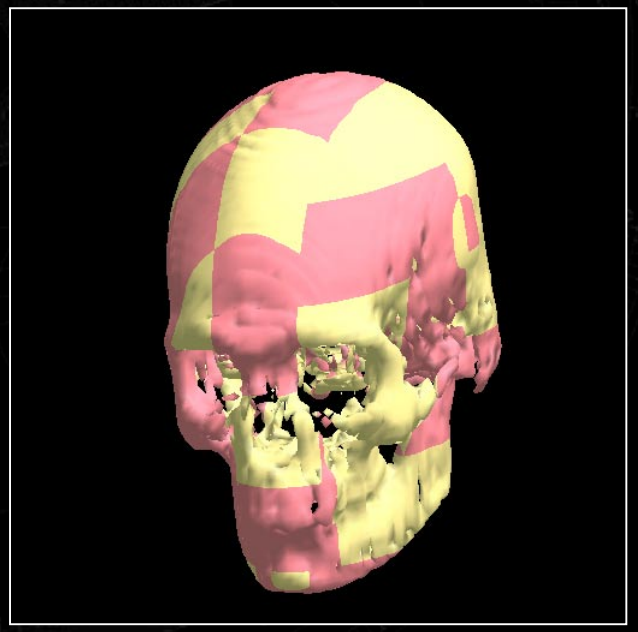
UNC/HP PixelFlow, Aizu VC-1, Stanford Lightning-2

Object Parallel

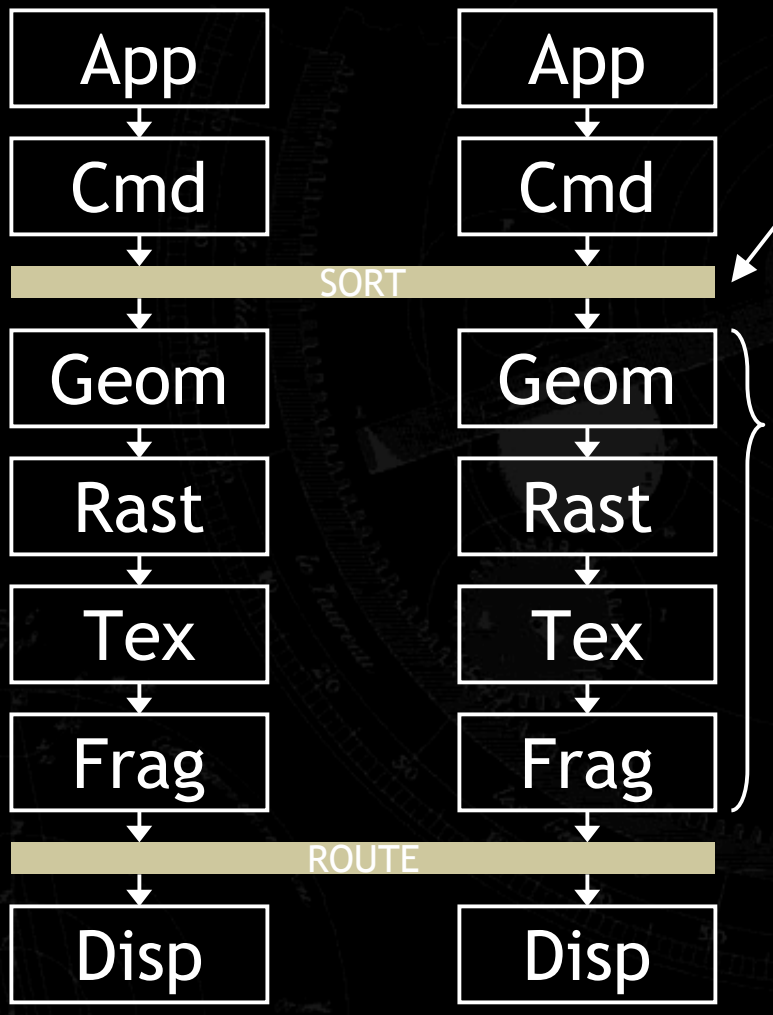




Z comp



Sort-First



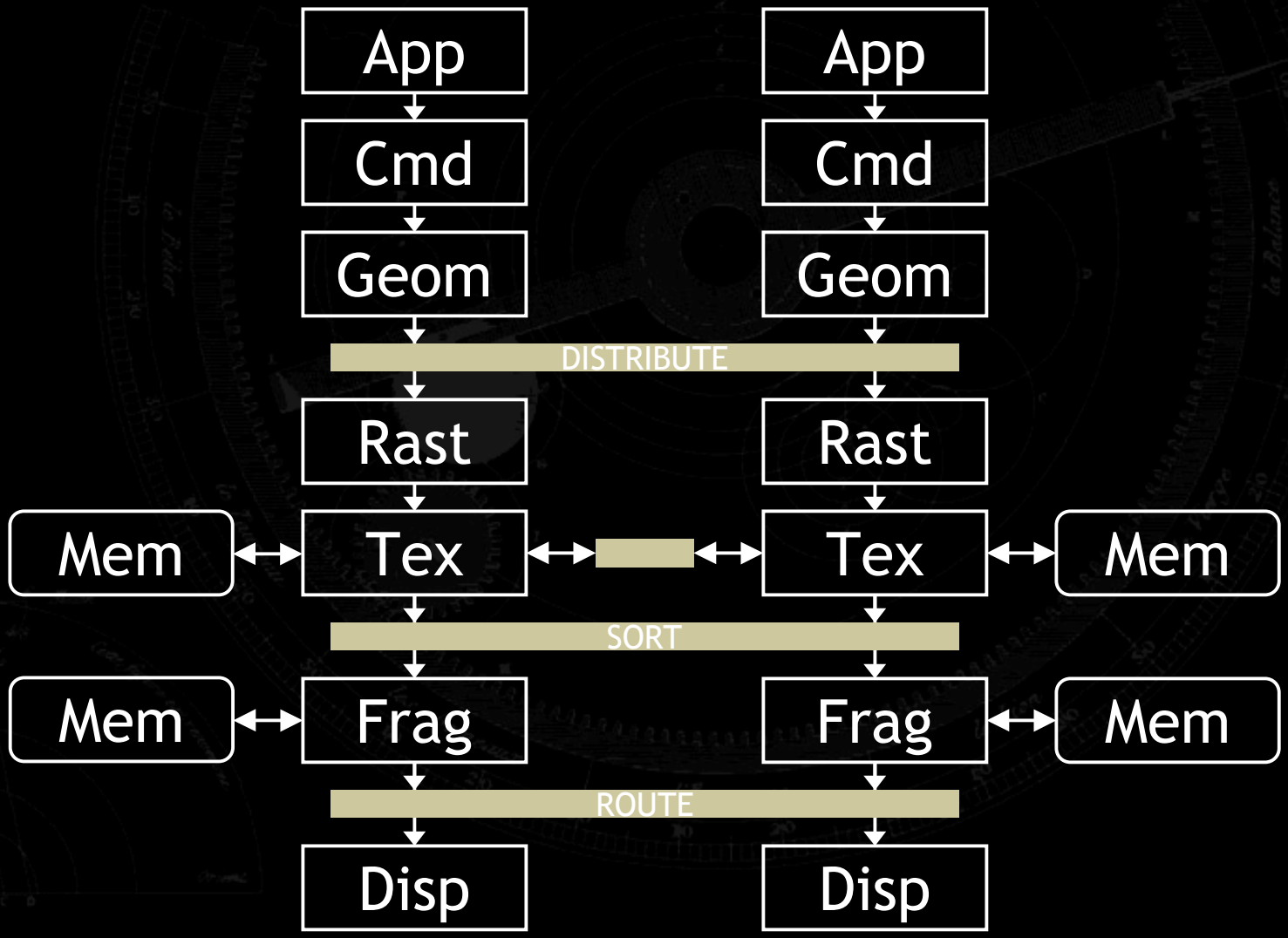
Point-to-point communication scales

Coarse tiling incurs *temporal load imbalance*

- **Matches capabilities of a single graphics card**
- **Maps well onto clusters with multiple displays**

Princeton Display Wall, Stanford WireGL

Sort-Everywhere: Pomegranate



Architecture Comparison

	Sort-First	Sort-Middle Tiled	Sort-Middle Interleaved	Sort-Last Fragment	Sort-Last Image Comp.	Pomegranate
Rasterization balanced	✗	✗	✓	✗	✗	✓
Scalable communication	✓	✓	✗	✓	✓	✓
Temporal load balance	✗	✗	✓	✓	✓	✓
Ordered	✓	✓	✓	✓	✗	✓

Summary

- **Graphics supercomputers (e.g. SGI IR/RM) not inherently scalable**
- **Clusters with commodity graphics cards inexpensive and potentially scalable**
- **Sorting/Routing/Distribution taxonomy**
- **Most promising short-term algorithms**
  - Sort-first for tiled displays
  - Sort-last with fast network (special network)
  - Sort-first tiled with image reshuffle
- **Parallel APIs critical for scalability**

Commodity-based Scalable Visualization: Graphics Cluster Components

Randall Frank

Lawrence Livermore National Laboratory

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

Scalable Rendering Clusters

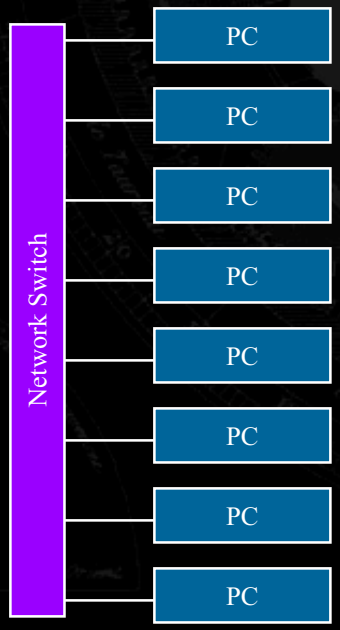
What makes a scalable rendering cluster unique?

- **Generation of graphical primitives**
  - Graphics computation: primitive extraction/computation
  - Multiple rendering engines
- **Video displays**
  - Routing of video tiles
  - Aggregation of multiple rendering engines
- **Interactivity (not a render-farm!)**
  - Real-time imagery
  - Interaction devices, human in the loop
- **I/O demands**
  - Access patterns/performance requirements

Graphics Cluster Anatomy: The Cluster

Start with a basic computational cluster

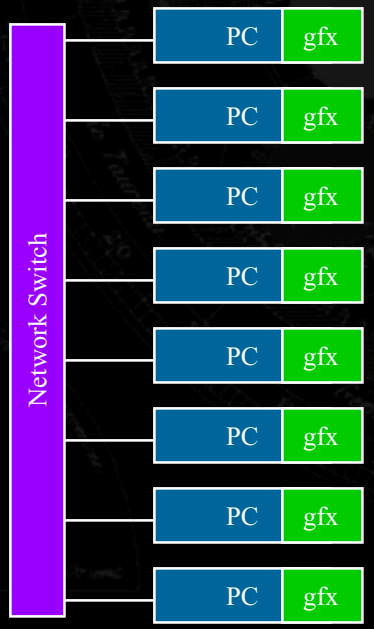
- COTS computational nodes
- High-speed interconnect
  - Gigabit Ethernet, Myrinet, ServerNet II, Quadrics,...



Graphics Cluster Anatomy: Rendering

Add multiple rendering resources

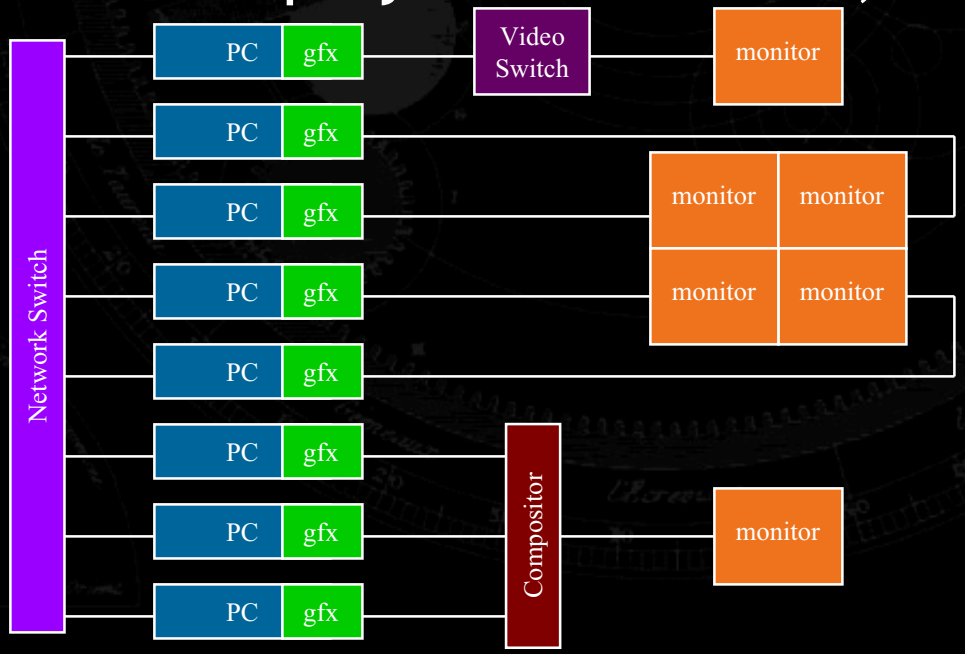
- Software rendering (mesa, custom, …)
- Hardware rendering cards
  - nVidia, ATI, 3dfx, intense3d, …



Graphics Cluster Anatomy: Displays

Attach one or more displays

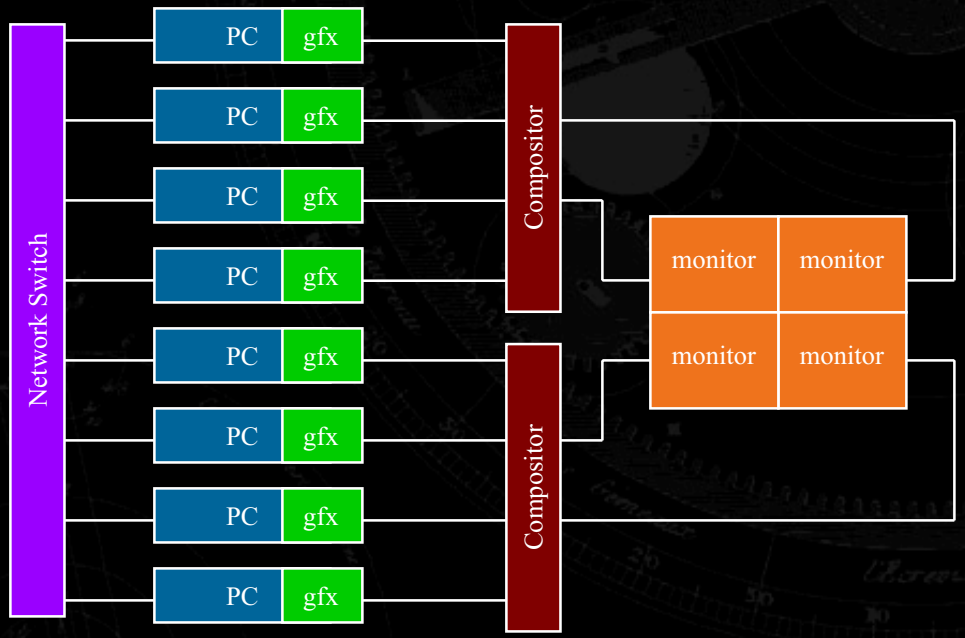
- Direct display monitors
- Tiled displays (PowerWalls)
- Composite displays: M renderers, N displays



Graphics Cluster Anatomy: Displays

Advanced layouts

- Combinations of tiling and compositing



LLNL PowerWall (6400x3072)

IBM Bertha (3840x2400)

PC Graphics Cards:What are they?

PCI and AGP commodity graphics cards

- PC architectures
- Intel and Alpha CPUs
- Common 3D Graphics APIs: OpenGL/DirectX

Why are we interested?

- Large numbers of cards - low cost
- Games + fast PC hardware - speed
- Graphics leadership

Broad categories

- Consumer - Games, Media playback
- Professional - CAD, Media generation

PC Cards: Consumer

Consumer: nVidia, ATI, 3Dfx, Matrox

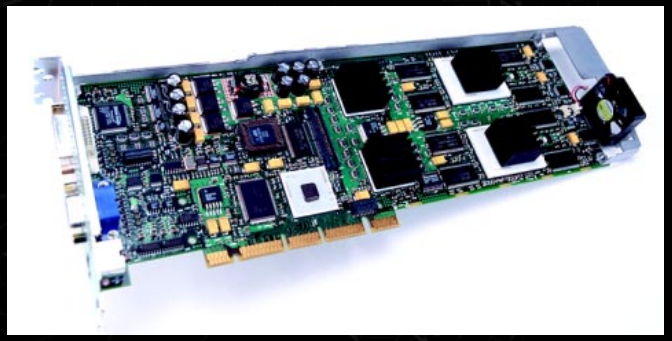
- Pros
  - High fill rates (400-1000Mpixels)
  - Hardware T&L (8-25Mtris) in most recent versions
  - Innovations: cube maps, texture combiners, vertex programs
  - Cheap (<$400), price sensitive/competitive market
- Cons
  - Driven by games
    - OpenGL can be a secondary consideration
    - Poor line drawing rates/quality
    - Windowing issues
    - Readback and buffer access issues
  - Difficult to achieve "ultimate" performance
  - Bit depth issues - good enough quality
    - Screen and pipeline (e.g. Texture compression)

nVidia GeForce 2

PC Cards: Professional

Professional: HP, IBM, 3DLabs/Intense3D, Vidia?

- Pros
  - Full accelerated OpenGL 1.2: 3D texture support
  - Finer attention to OpenGL detail
  - Deeper intermediate computations
  - Non-game features
    - Higher line drawing performance
    - Larger memory
    - Concurrent multi-bit depth/screen support
    - Texture download performance



3DLabs Wildcat II 5110

- Cons
  - Lower fill rates (100-200Mpixels, application market bias)
  - Fewer "innovative" extensions: Cube mapping
  - (More) Expensive

PC Cards: What should you expect?

- Are they really Infinite Reality™ pipes?
  - Basic rendering and raw speed: for most measures, yes
  - Image quality/integrity: no, improving
  - Flexible output options: no + DVI, improving, but no DG5-8s
  - System bandwidths: maybe
- Easily rival present desktop workstation graphics
  - Vendors are shipping them as options
- System stability issues (Read the game torture test reviews)
- High fill rates (Not high enough, thank the BSP tree)
- Future feature sets
  - Exceed the IR in many ways, can be raw and complex
  - Extensions: increase the difficulty in writing portable code

Graphics Cluster Anatomy: Issues

- **System bus contention**
  - Simultaneous graphics AGP bandwidth and interconnect PCI bandwidth
  - Careful selection of motherboards (e.g. i840)
- **CPU options (number/speed)**
  - System overhead (e.g. TCP/IP stacks)
- **Core system interconnect**
  - Bandwidth/latency
- **Operating system selection**
  - Drivers/cluster management software

Aggregation: Tiling Vs Compositing

Goal: aggregate multiple rendering engines, combining their outputs on a single display to scale rendering "performance"

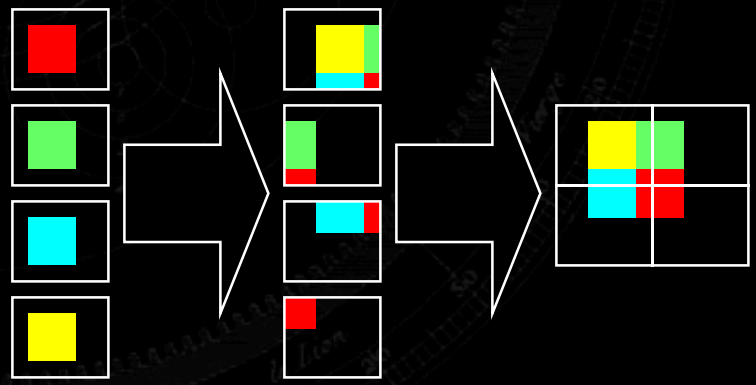
- **2D - "screen space"**
  - "Sort-first" rendering model
  - Targets display scalability, higher frame rates
- **3D - "data space"**
  - "Sort-last" rendering model
  - Targets large data scalability, higher polygon counts

Aggregation: Tiling

Tiling (2D decomposition in screen space)

Route portions of a final aggregate display to their final destination with no overlap

- Order independent
- Destination determines bandwidth
- Graphics primitives may be moved, replicated or sorted for load balancing
- RGB data

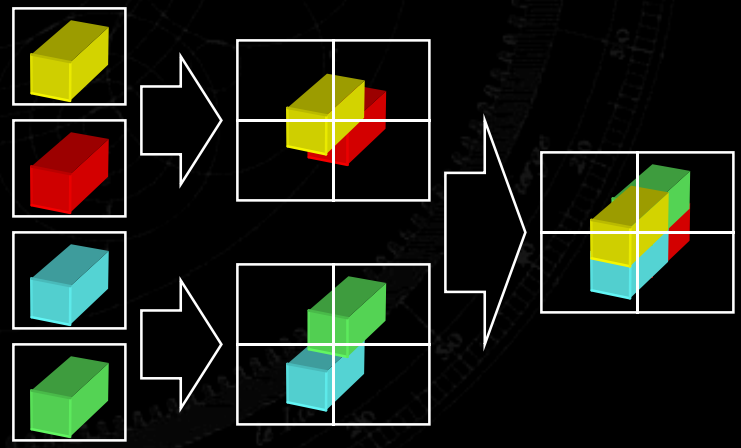


Aggregation:Compositing

Compositing(3D decomposition in data space)

3D blocks that are combined using classic graphics operators (e.g. Z-buffering, alpha blending, etc)

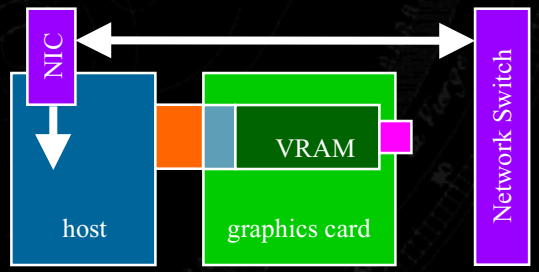
- Z, α, stencil enhanced pixels
- Fixed 3D data decompositions (data need not move)
- Bandwidth exceeds that of output display (3D vs 2D)
- Hierarchy trades bandwidth for latency
- Ordering may be critical



Implementing Aggregation

Compositiondatapathsare targets for specialized parallel and asynchronous interconnects

- **Basic operation**
  - Access the rendered imagery in digital form
  - Route image fragments to composition mechanism
  - Composite the fragments
  - Display the results
- **Approaches**
  - Reuse the cluster interconnect
  - Utilize digital video interface (DVI) output
  - Use a dedicated interconnect



Reuse Core Cluster Interconnect

Compositing/tiling directly on the nodes

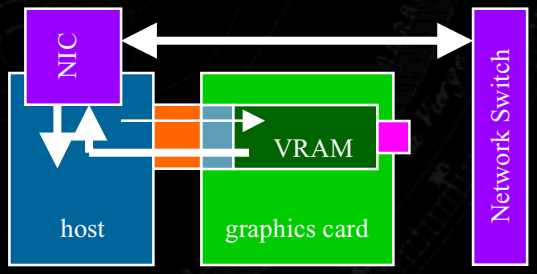
- Image or primitive exchange over the interconnect
- Readback of graphics card buffers (RGB,z,α,stencil)
- Flexible computation of aggregate imagery by host CPU

Current solutions

- Quadrics, Myrinet, ServerNet, GigE
- MPI, VIA, TCP/IP,GM

Issues

- Processor overhead (second CPU?)
- Available bandwidth and latency
- Framebuffer readback performance



Myricom Myrinet 2000

Digital Video Interface Interconnect

Video based solutions

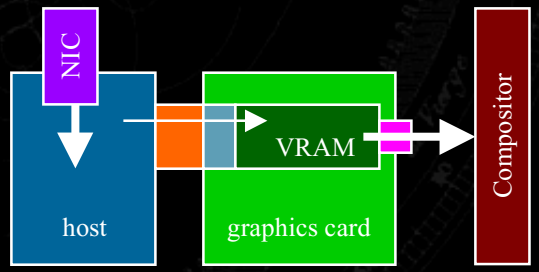
- Ideally suited to tiling, DVI inputs/outputs
- Asynchronous operation, Avoids readback

Lightning-2

Examples

- Stanford: Lightning-2, Texas: MetaBuffer

Issues



- Synchronization issues
  - Tagged imagery
  - Auxiliary signals
- Limitations of DVI signal and pixel formats
- Limited compositing functions/ordering options
- Scalability of mesh architectures

Dedicated Compositing Interconnect

Secondary interconnect dedicated to compositing

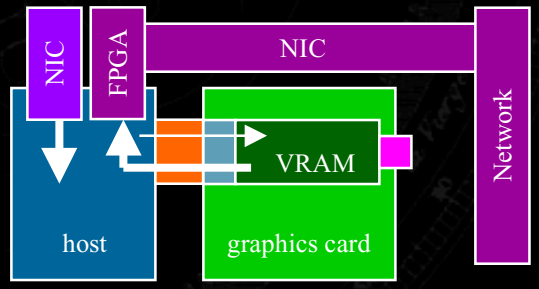
- Need not be fully connected (data decomposition)
- Offload operation from host onto custom chips (FPGA)
- General pixel formats, programmable composition functions
- Interconnect switch for ordering

Examples

- Compaq: Sepia, IBM: SGE

Issues

- Framebuffer readback
- Additional host bus demand
- Bandwidth-pixel count/format



Sepia-2

Composition and Interconnects: Issues

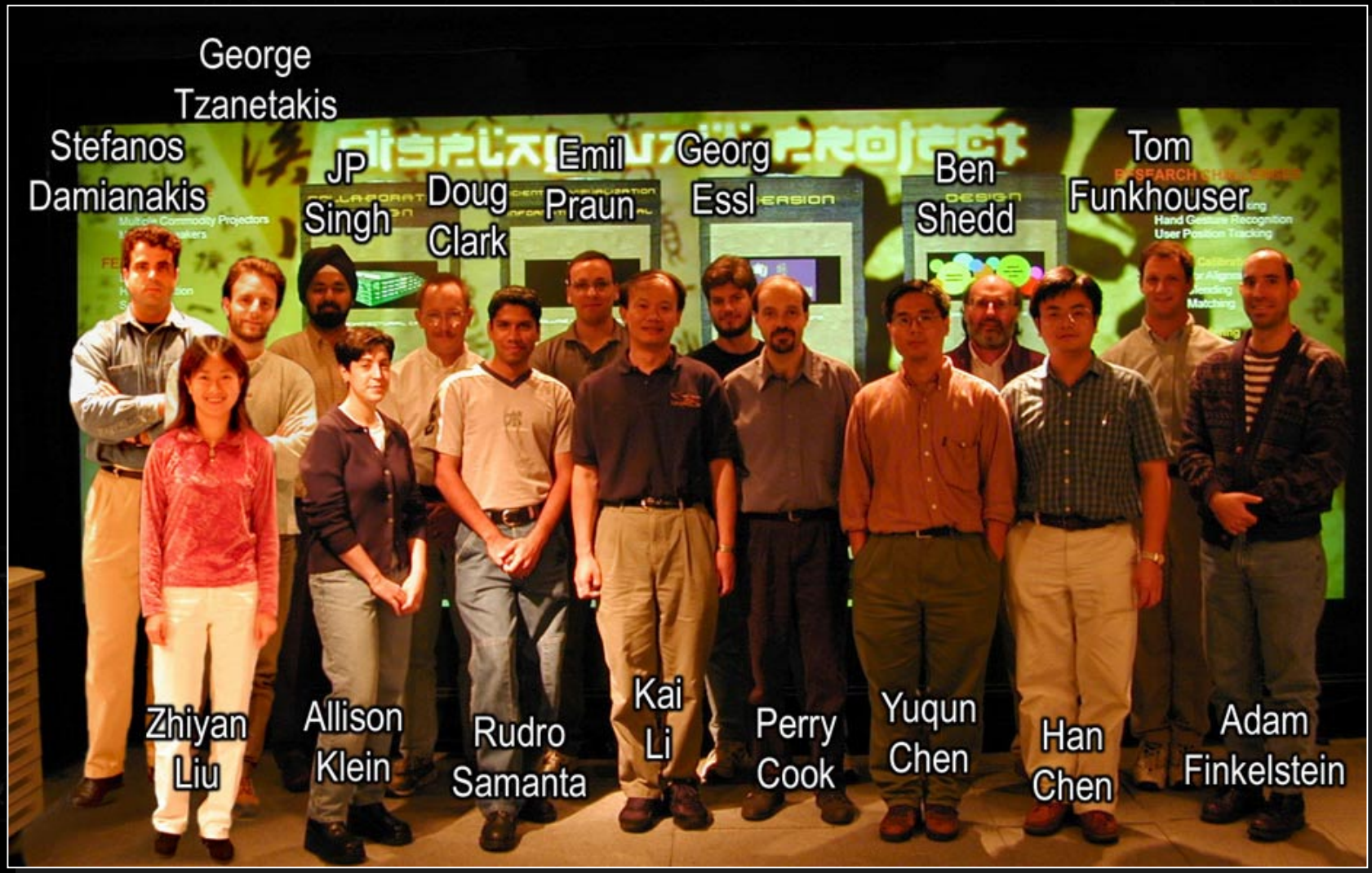
- Multi-pass rendering algorithms
- Framebuffer readback
  - Performance and availability of graphics APIs
  - Limitations of DVI: distance, pixel formats, bandwidth
- Graphics card bit depth limitations (e.g. global Z)
- Latency and ultimate framerate issues
- Protocol/API inefficiencies
  - TCP/IP: High overhead, Jumbo frames
- Flexible/scalable software interfaces
  - The “zoom” problem
  - Data partitioning

Challenges in Building Tiled Displays

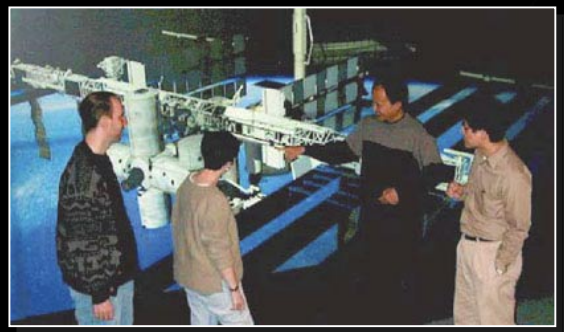
Adam Finkelstein
Computer Science
Princeton University

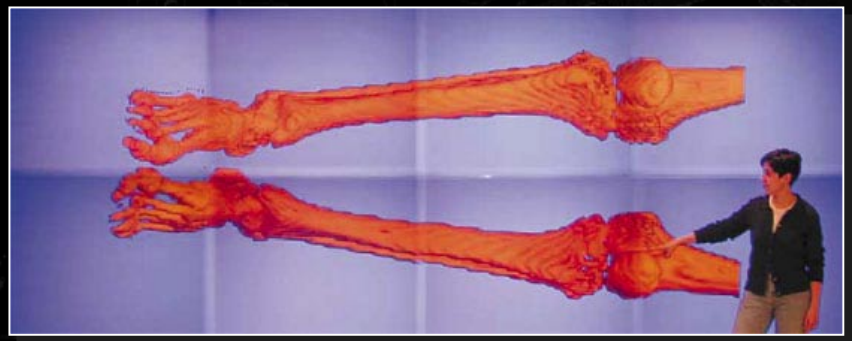
SIGGRAPH 2001
EXPLORE INTERACTION AND DIGITAL IMAGES

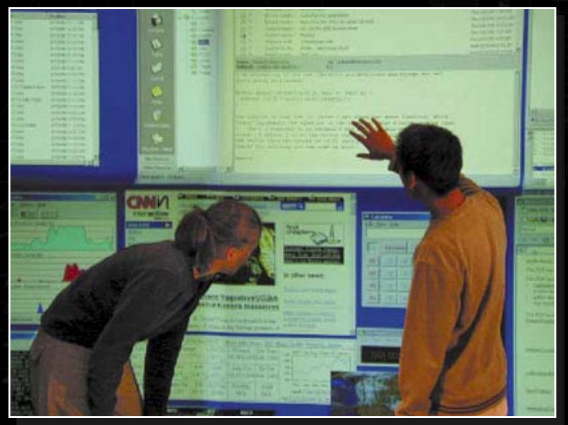
Princeton Display Wall Team

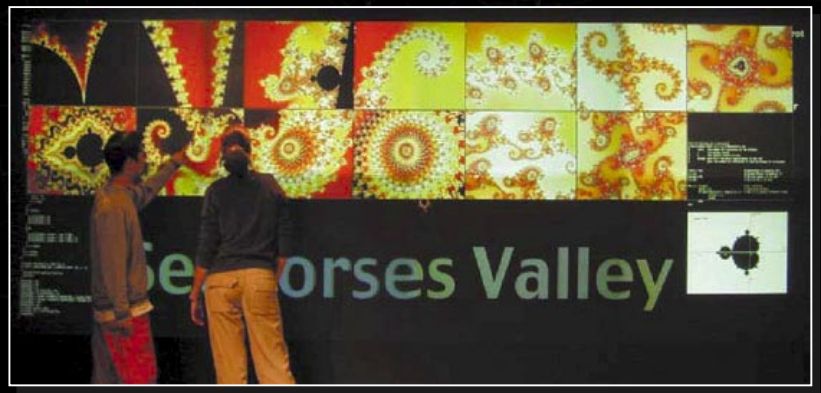


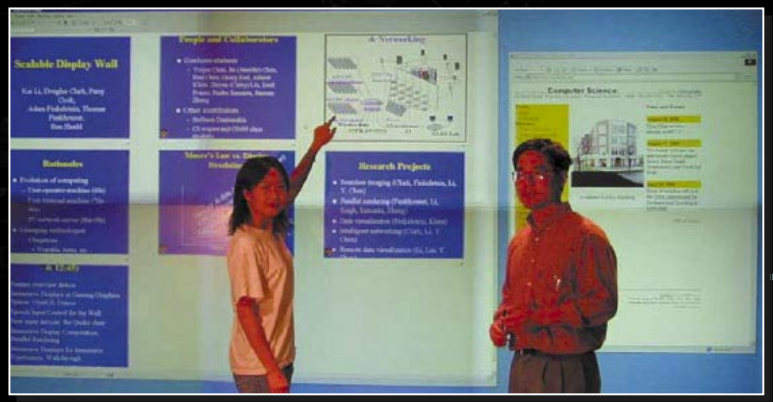
Motivation








orses Valley


Scalable Display Wall
Rationales
Research Projects
Computer Science

Why tiled displays?

Ideal for data visualization

- High resolution, human scale, broad dynamic range

Single display devices are inadequate

- Data is growing faster than resolution
- Compute power is growing faster than resolution

Tiling: Use multiple display devices as one

- Challenge I: quality
- Challenge II: scalability

Displays (not projected)

First generation

- CRT: Excellent color, no tiling, inefficient

Second generation

- LCD: Nice color, only four tiles, efficient
- Plasma: Nice color, no tiling, inefficient

Third generation (future)

- OLED: excellent color, tiles ok, flexible, very efficient

Projectors

First generation

- CRT: Excellent color, tile ok, not bright

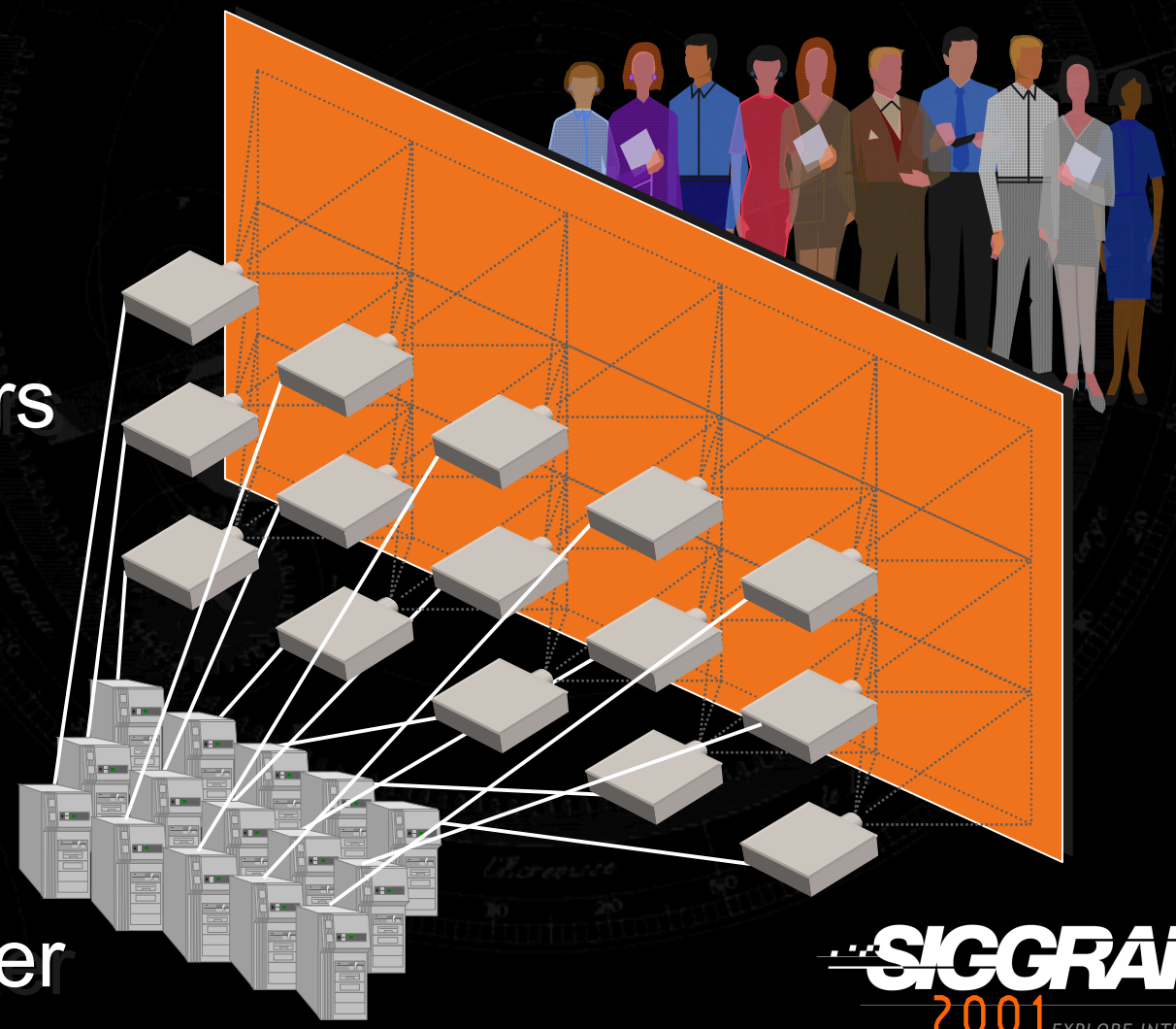
Second generation

- LCD: Very bright, bad color temperature
- DLP: Quite bright, nice color, expensive

Third generation

- Laser: Possibly a contender,
  … but price must drop

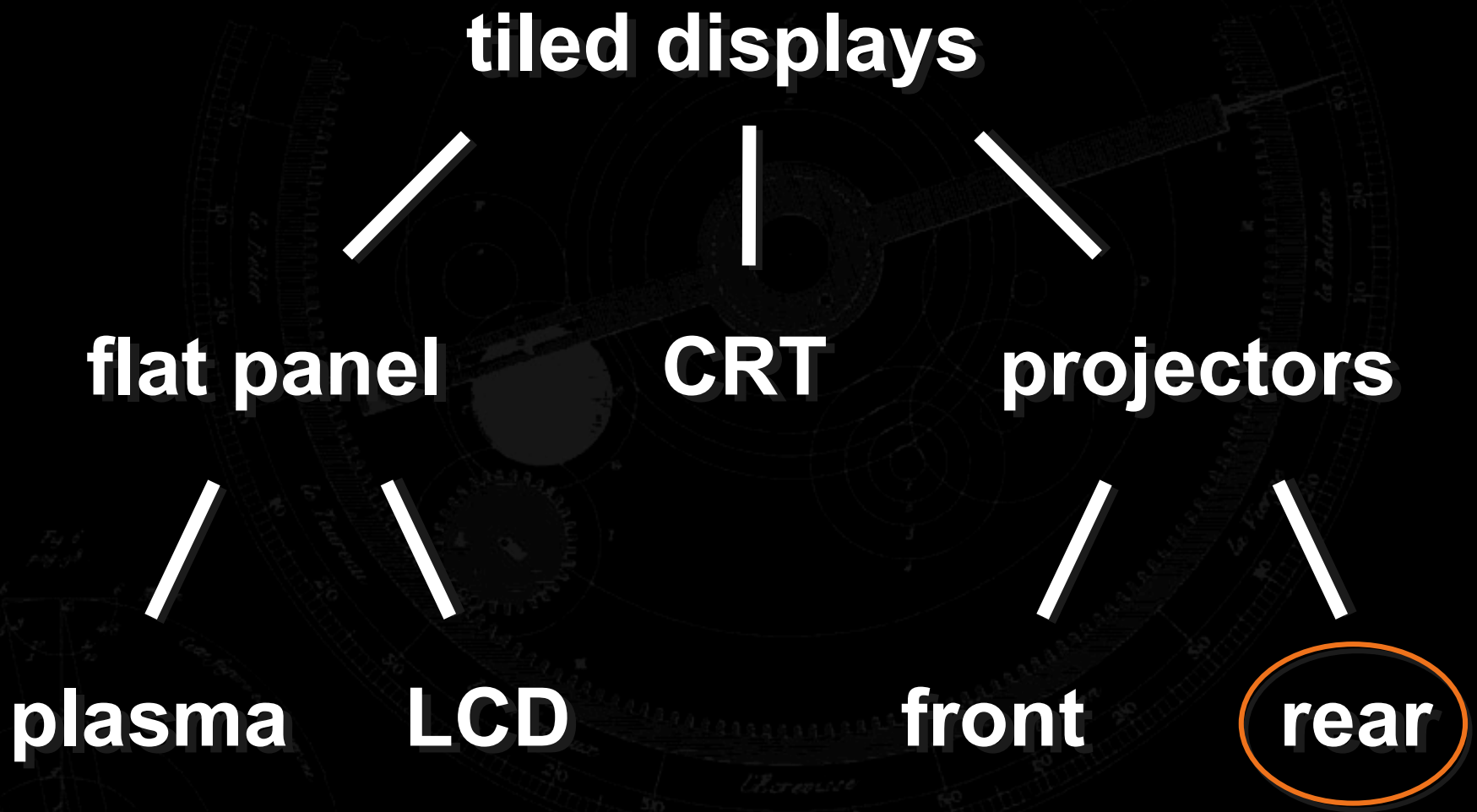
Rear projection



projectors

PC cluster

Taxonomy



Princeton Display Wall

Seamless Display

- Geometric alignment
- Blending
- Color balance

Geometric Alignment

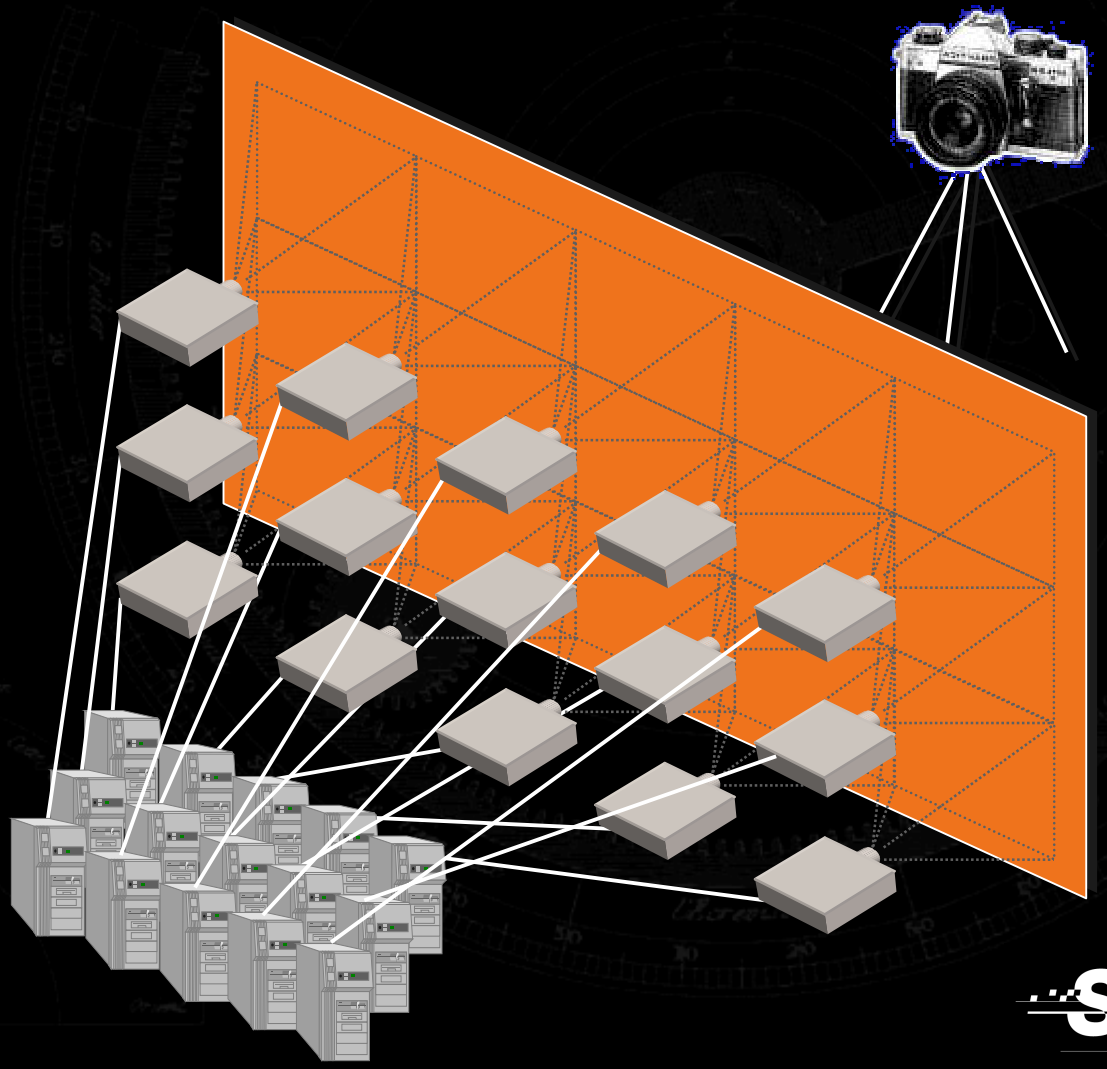
Perfect alignment is difficult

- Many degrees of freedom
- Lens distortions

Solutions

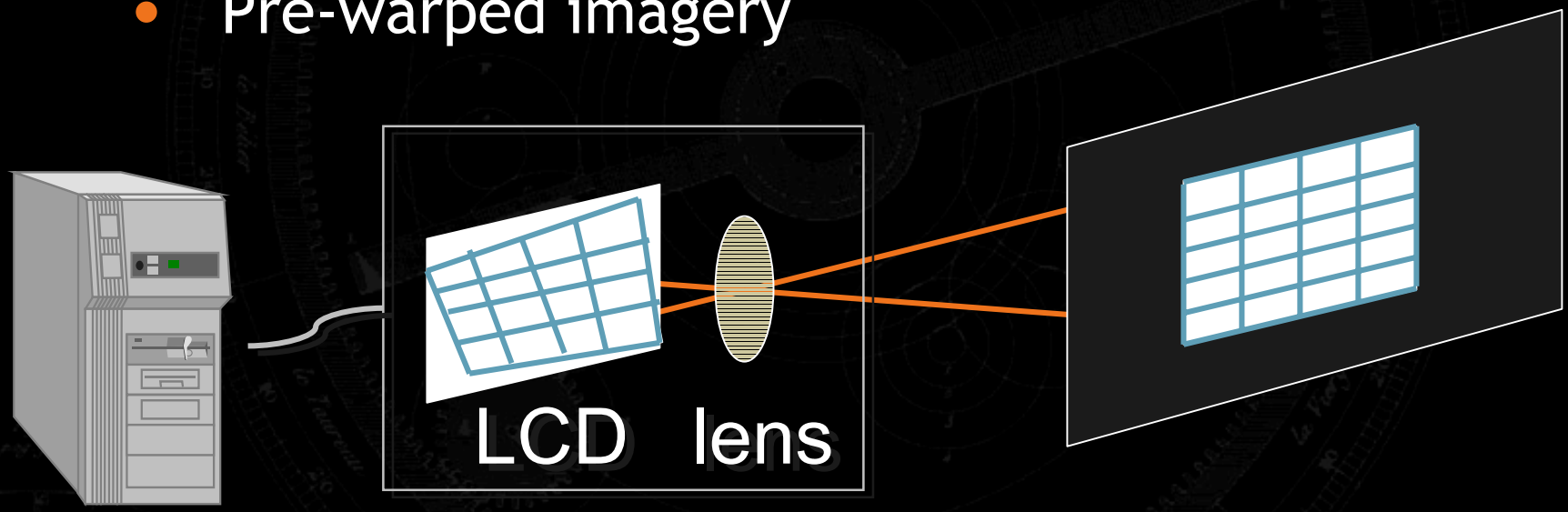
- Calibrated camera (UNC and MIT)
- Uncalibrated camera (Princeton)
  $\rightarrow$ Uses global optimization

Geometric Alignment [Chen2000]

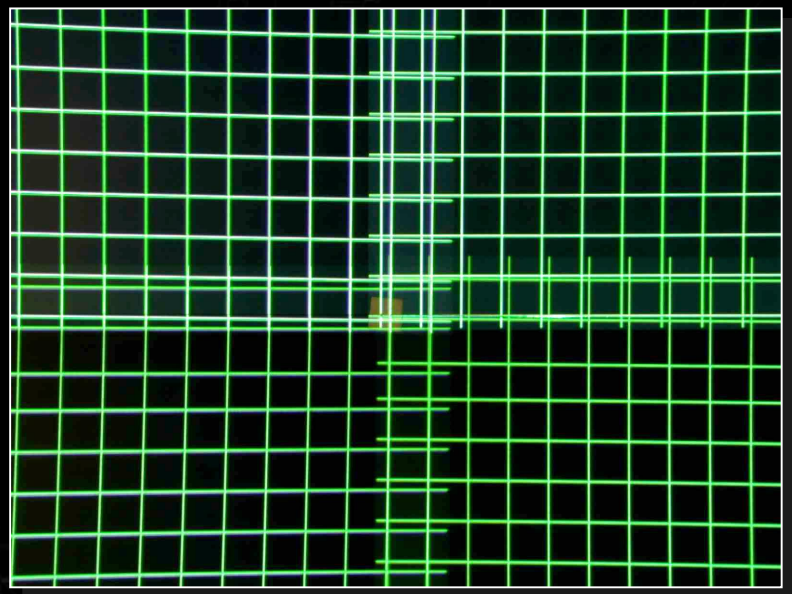


Geometric Alignment

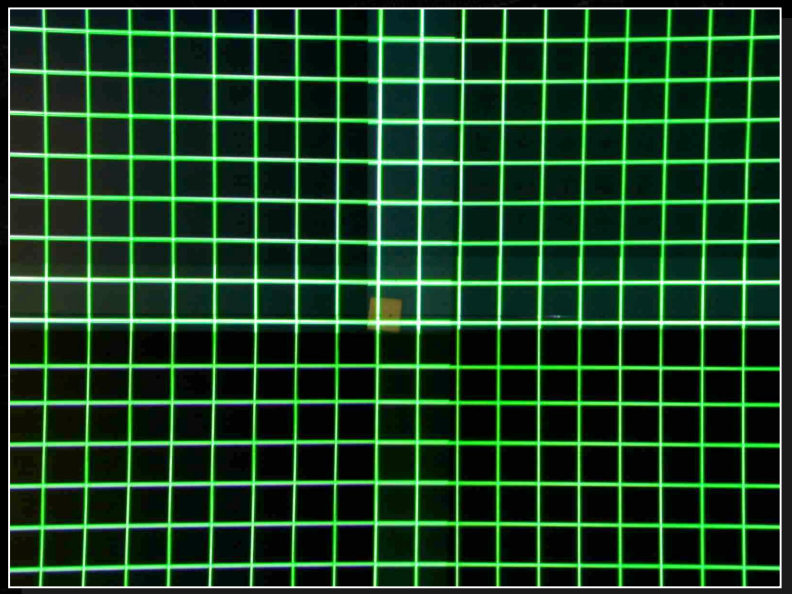
- Pre-warped imagery



Geometric Alignment



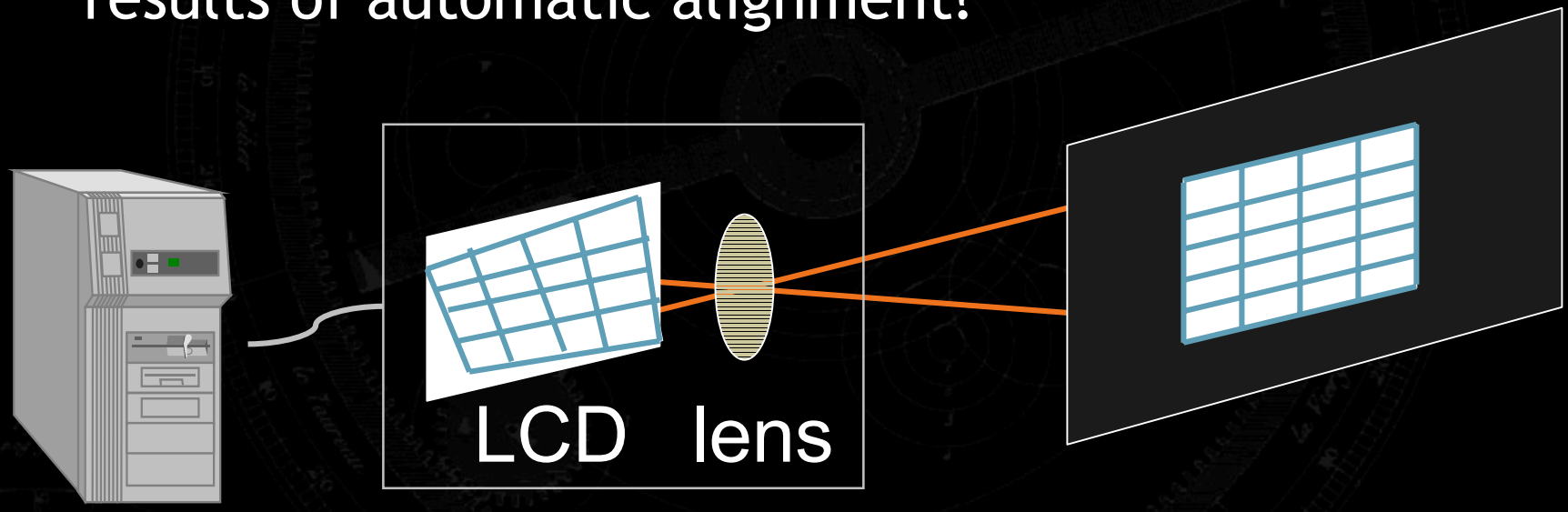
before



after

Geometric Alignment

Challenge: how do we use the results of automatic alignment?



PC **projector** **screen**

Geometric Alignment

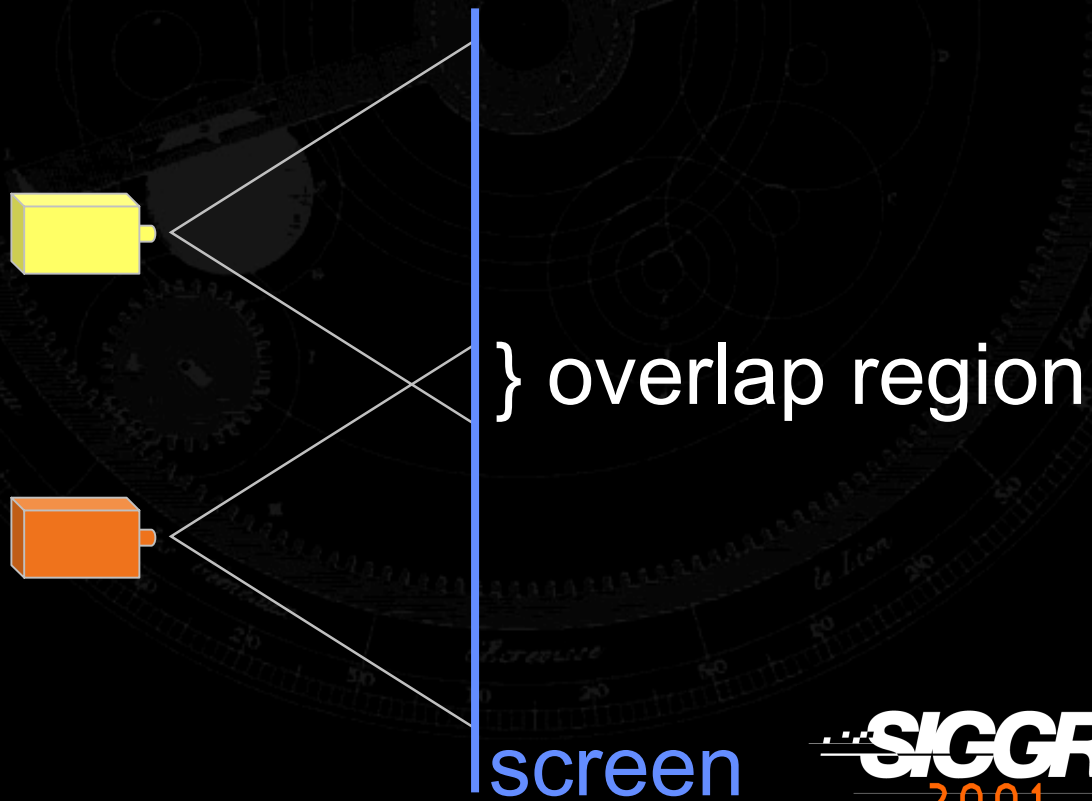
Challenge: how do we use the results of automatic alignment?

Still images: use texture mapping

3D-models: use a matrix transform

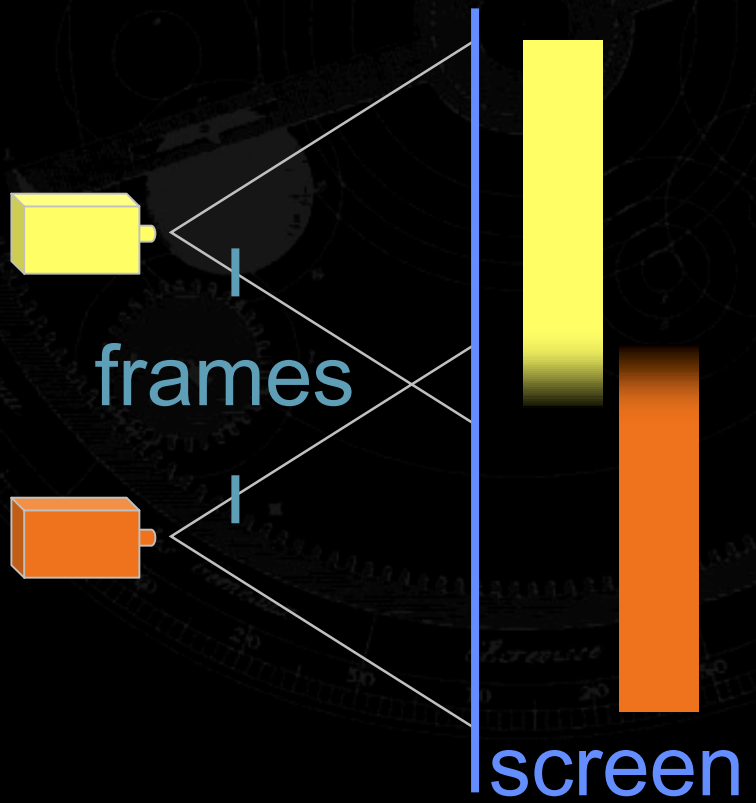
MPEG video: decode pixels block-by-block

Blending

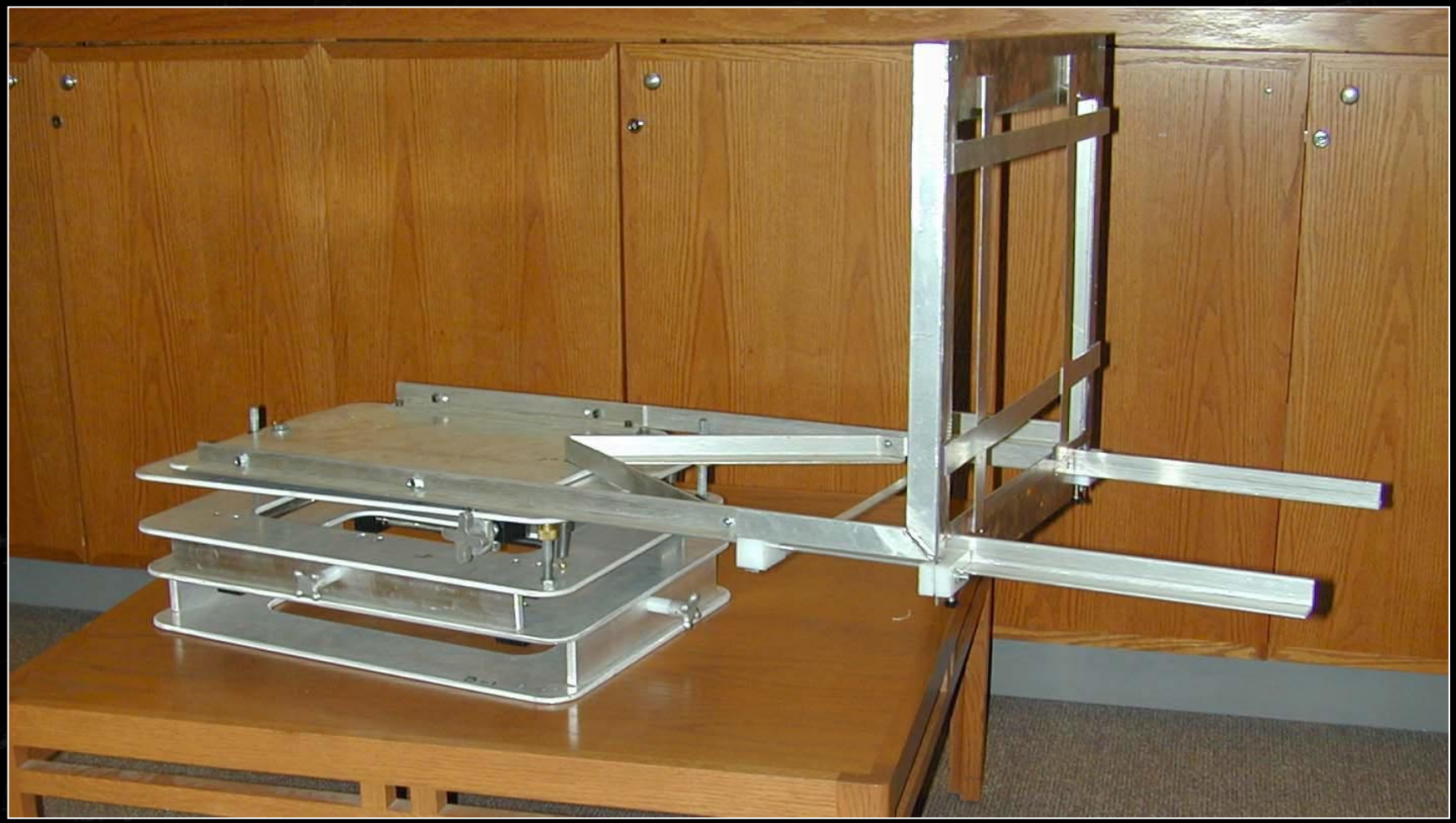


Blending

Problem: black + black = gray



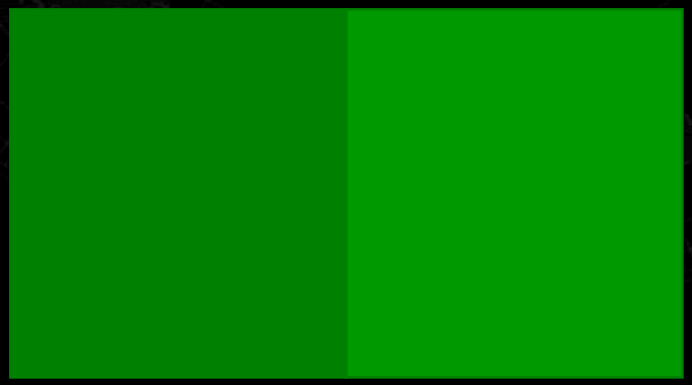
Blending: projector mount



Color Balance

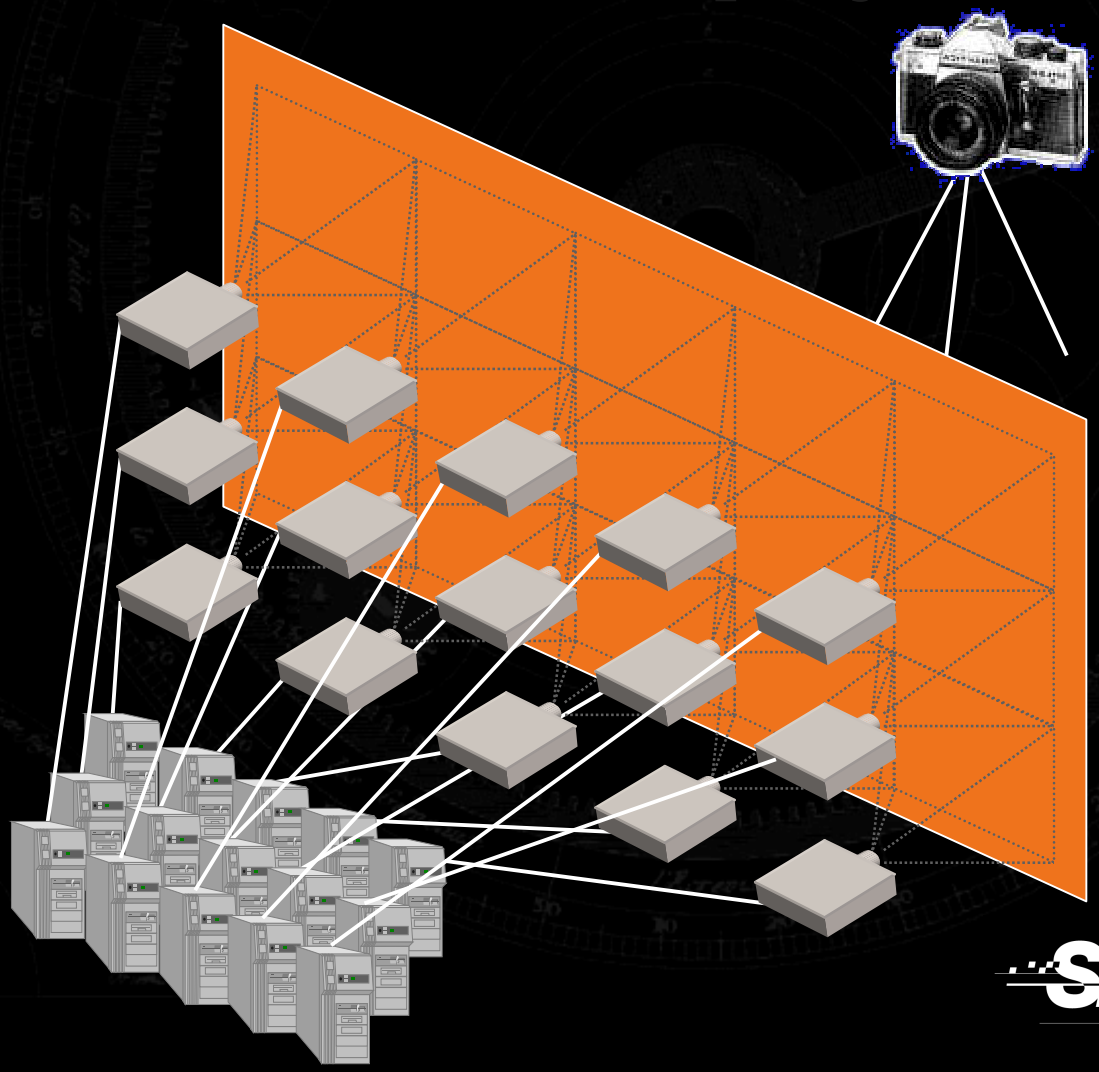
Projectors produce different colors

- Different lamps
- Lamps degrade at different rate
- Wide variation, even in same model



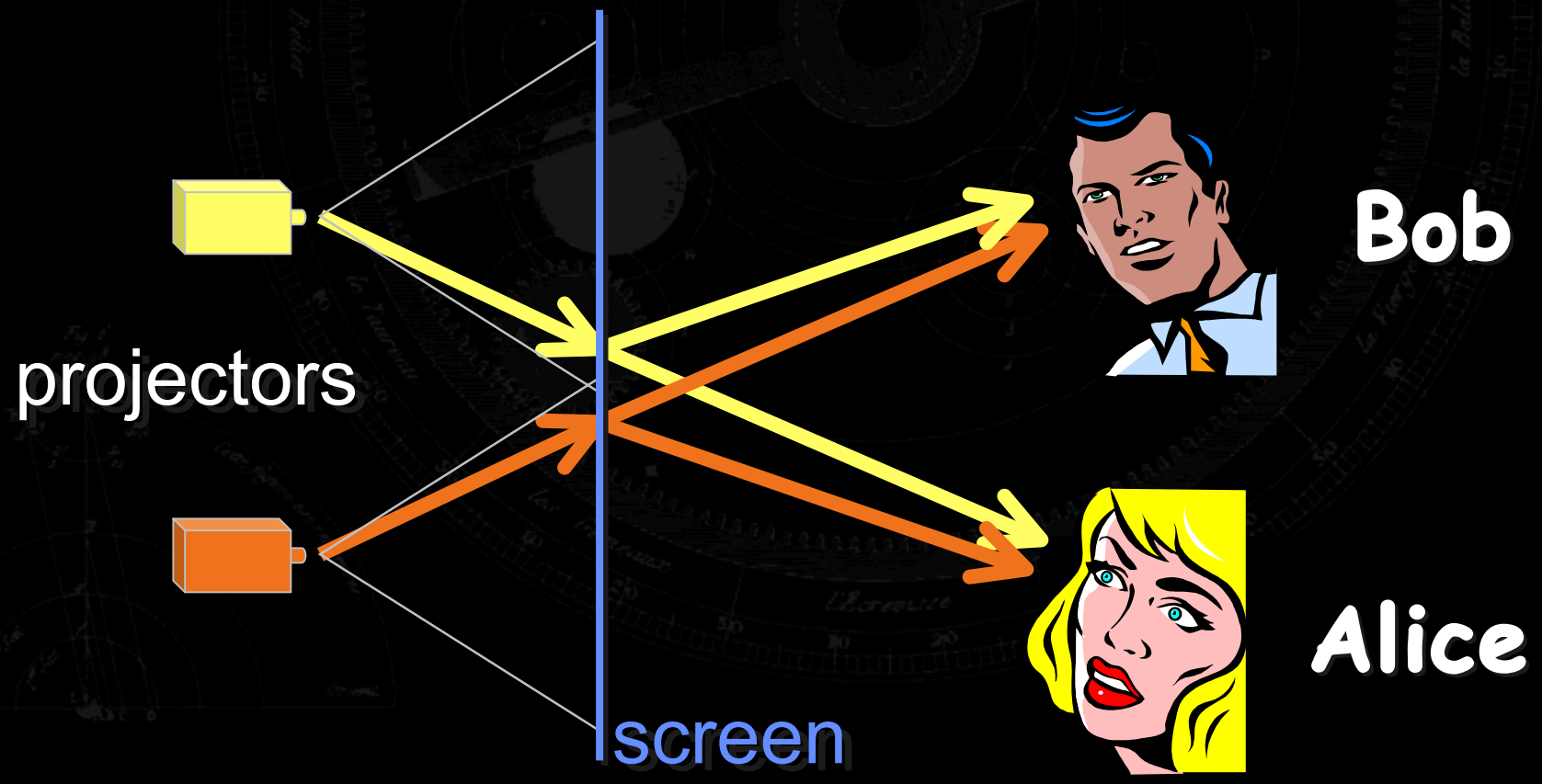
Color Balance

[Majumder2000]



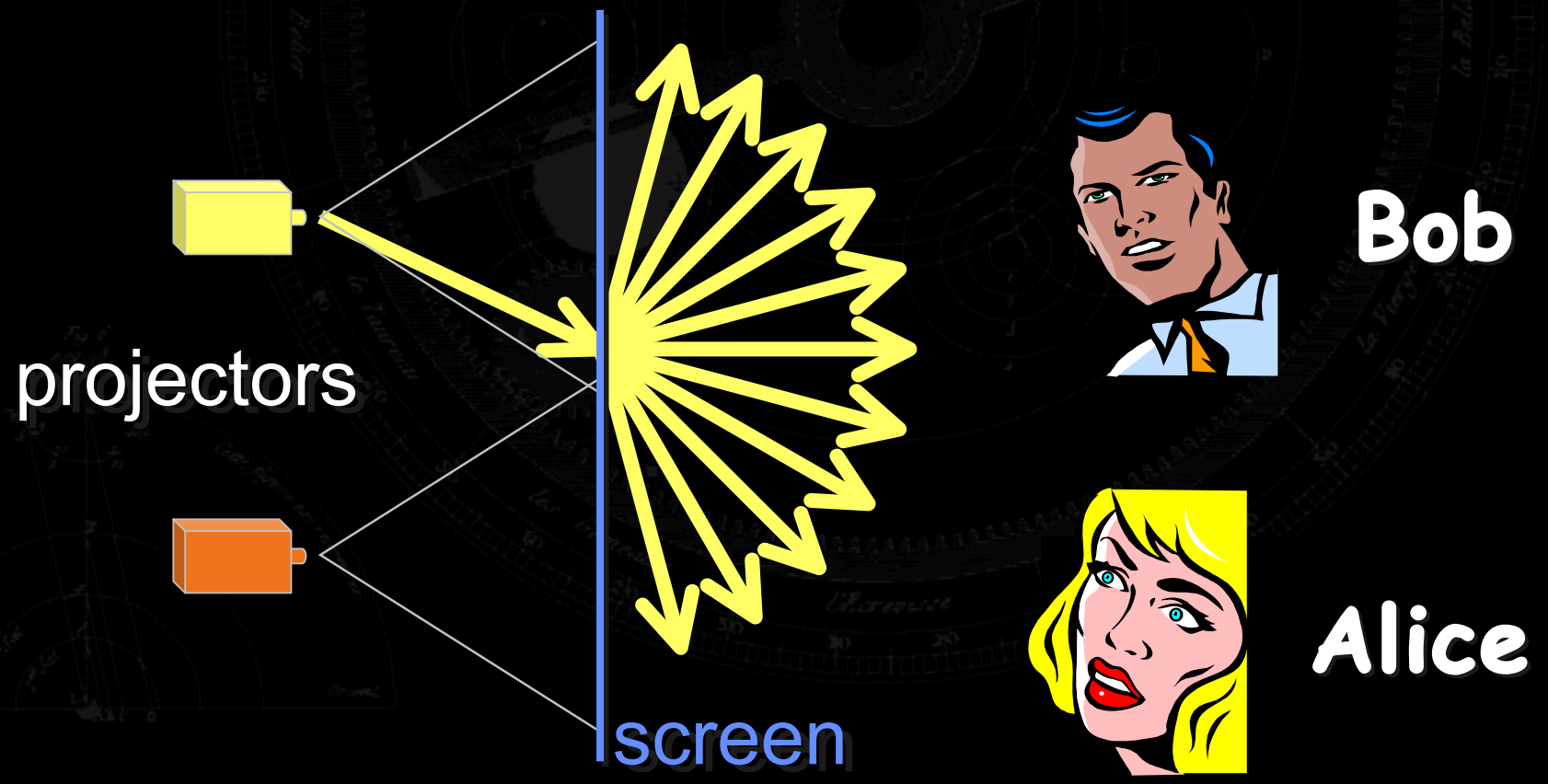
Color Balance

Challenge: often view-dependent



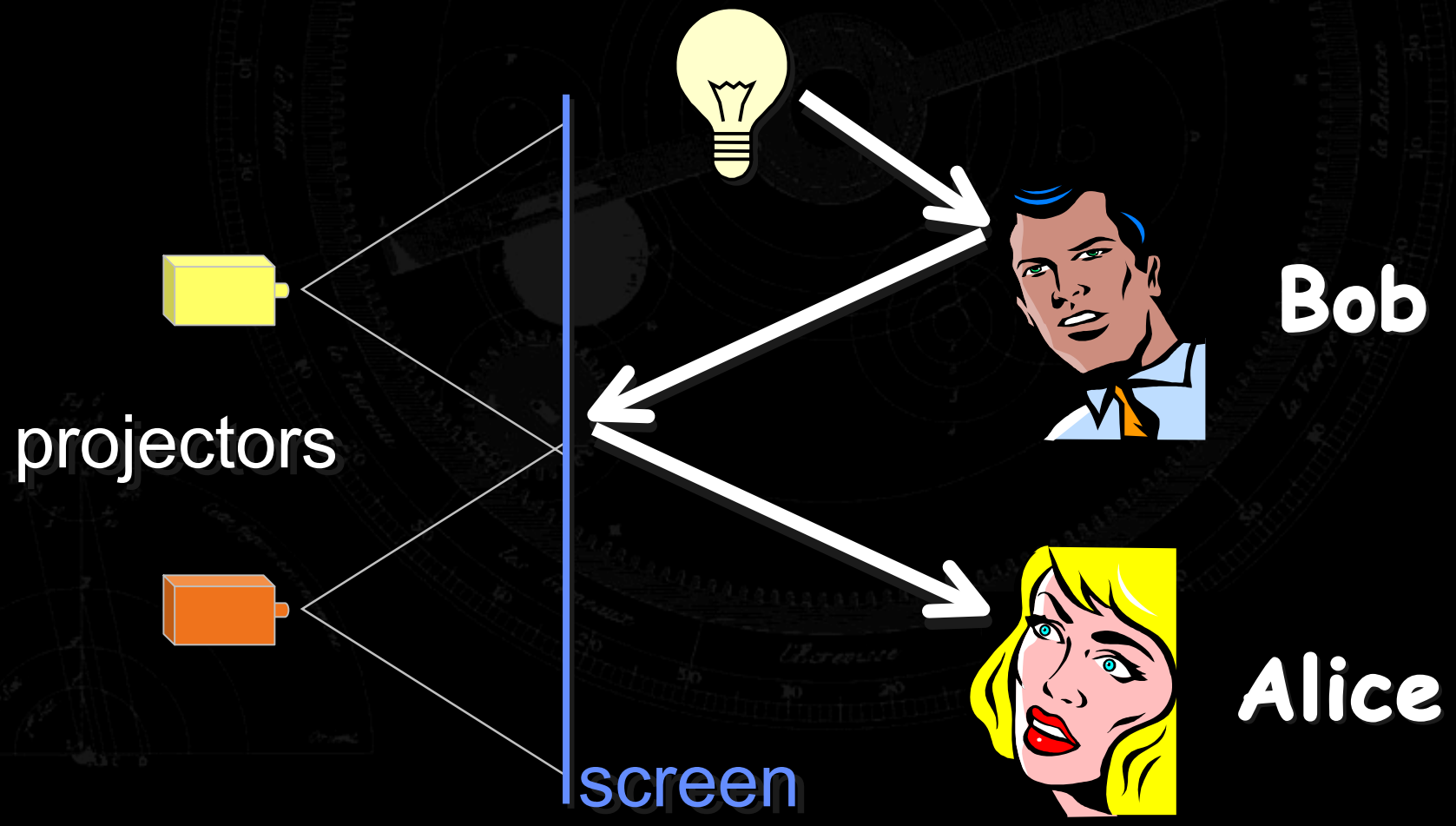
Color Balance

Challenge: often view-dependent



Color Balance

Challenge: often view-dependent



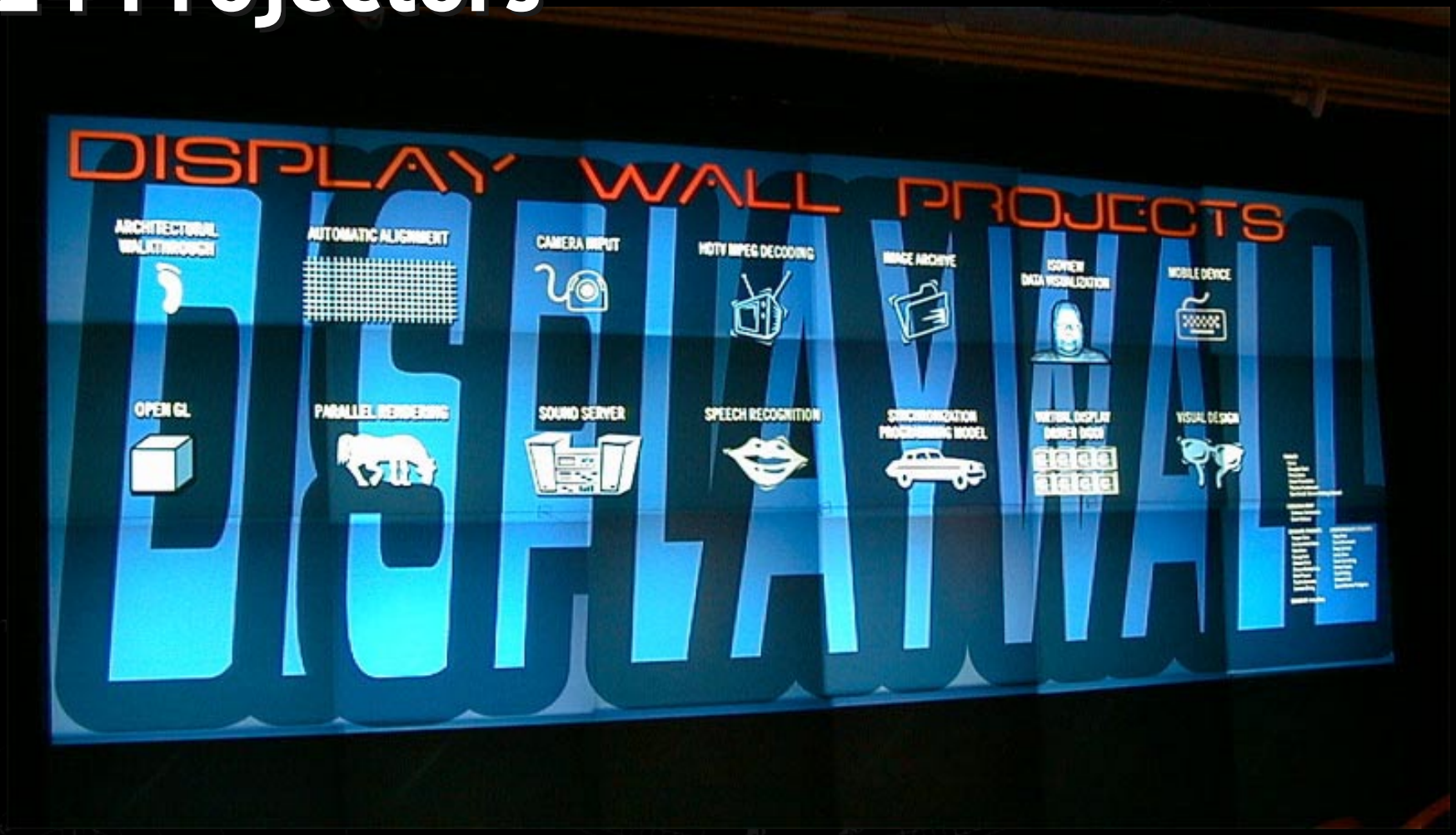
Color Balance

Challenge: often view-dependent

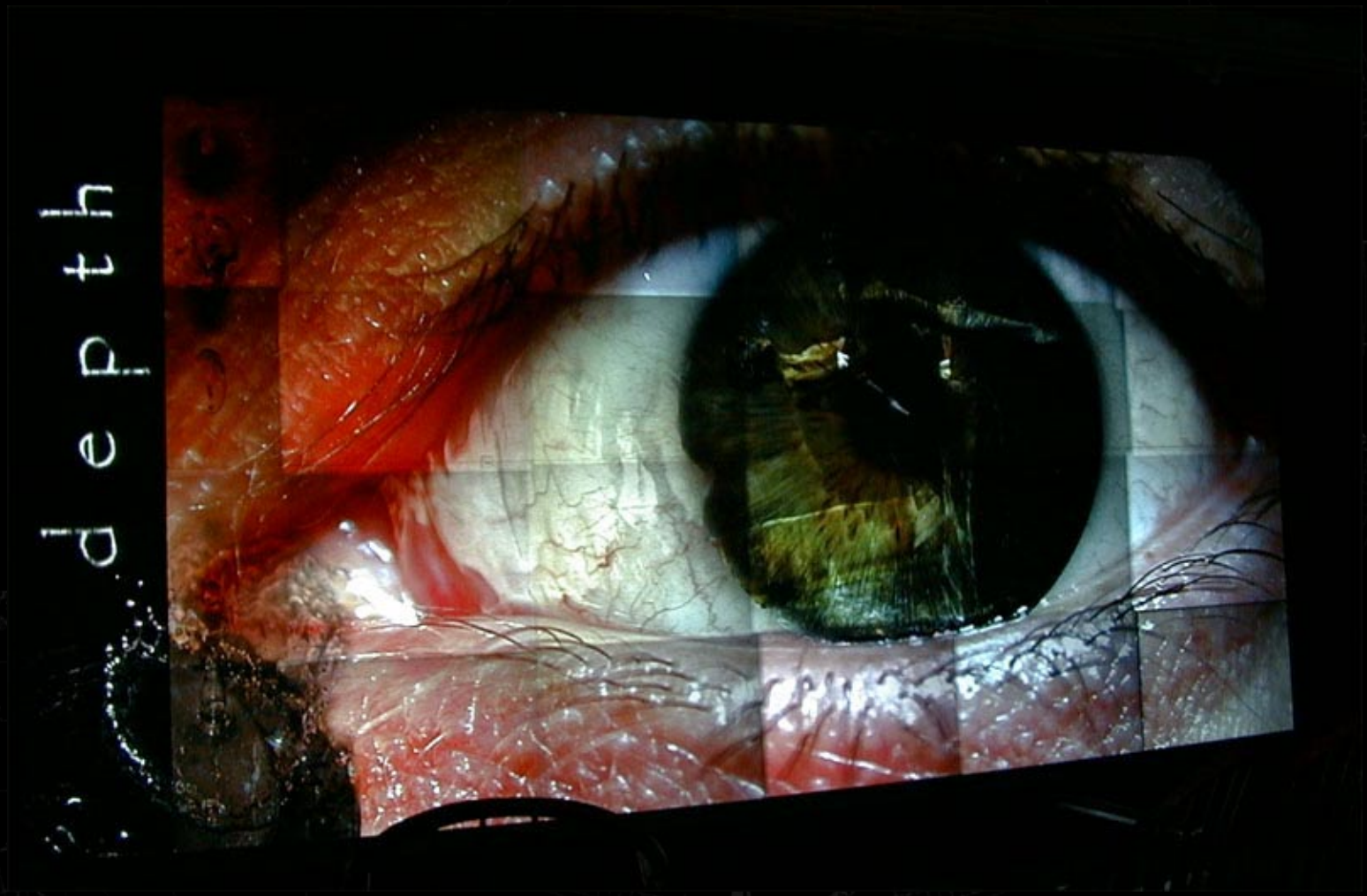
The ideal screen:

- ✓ Transmissive
- ✓ Diffuse
- ✓ Black

24 Projectors



24 Projectors



Campus center installation

Summary

Commodity components drive design.

- Tiling projectors is a viable approach for scalable, high resolution display
- Can build an inexpensive display wall
- Seamless tiling remains a challenge

DOE/ASCI-Lab Research Efforts @ Sandia National Laboratories

Constantine "Dino" Pavlakos

Scalable Rendering Team, Sandia National Laboratories

Brian Wylie, Ken Moreland, Vasily Lewis, David Shirley, Milt Clauser, Carl Diegert, Dan Zimmerer, Carl Leishman, Jerry Friesen, Jeff Jortner

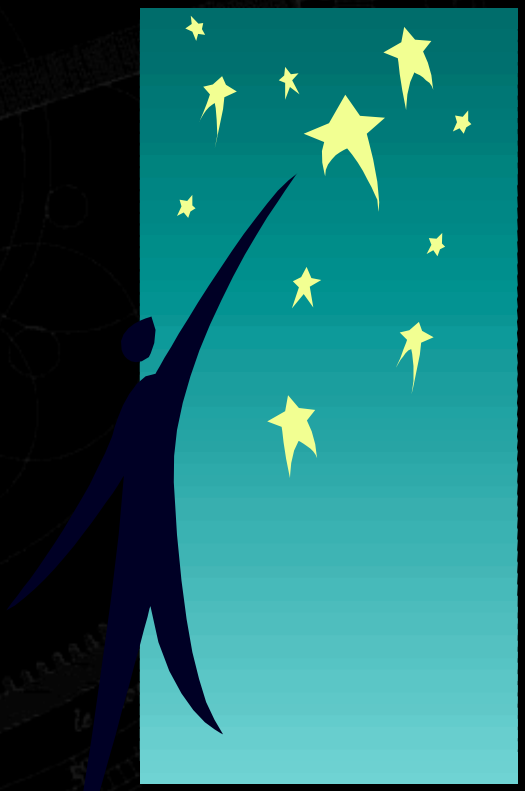
Sandia National Laboratories

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

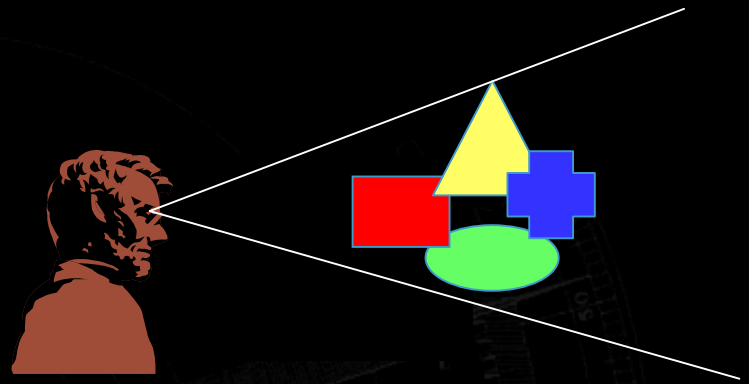
Early efforts - “Horizon” Cluster

- Sandia’s early R&D graphics cluster
- 16 SGI 320’s 450Mhz PIII, 512MB RAM
- Cobalt Graphics [1.75Mtri/sec with Vertex Arrays]
- Unified Memory Architecture (no special code developed)
- Windows 2000
- GigE Interconnect [Foundry Switch/Alteon NIC’s]
- Poor interconnect performance [270Mb/sec, no jumbo packets]
- Cost was ~$200K

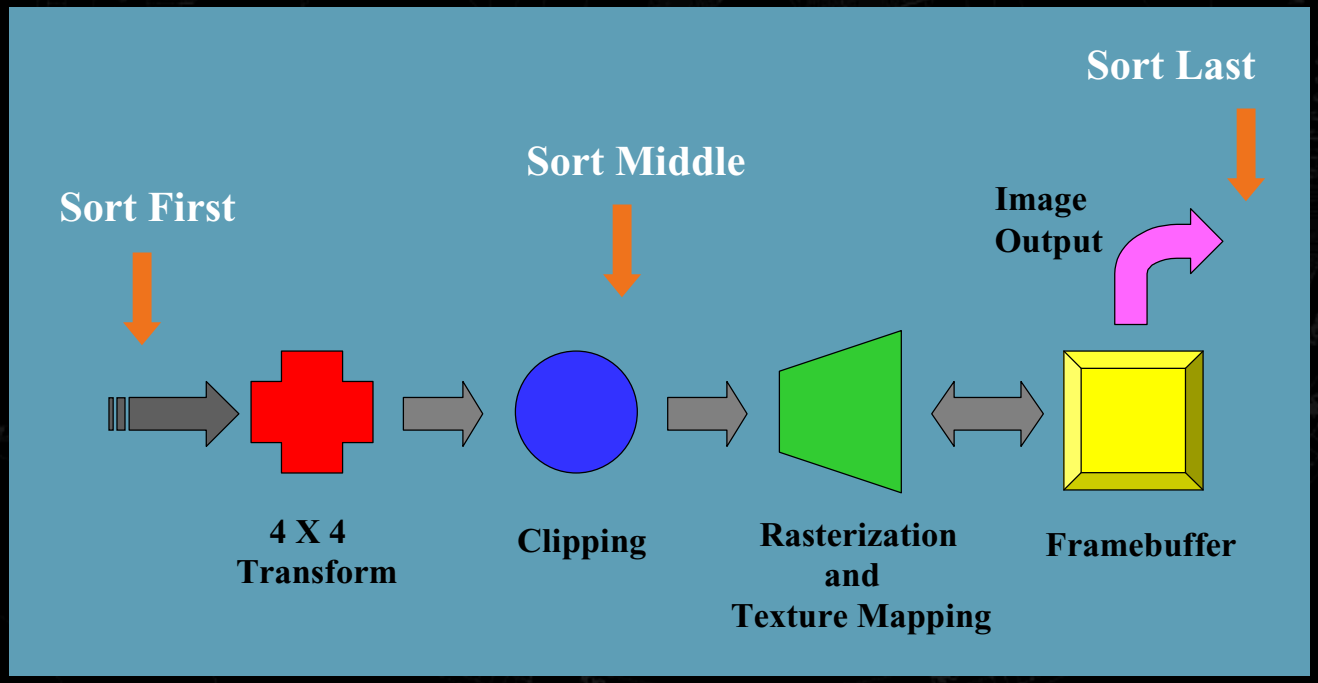


SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

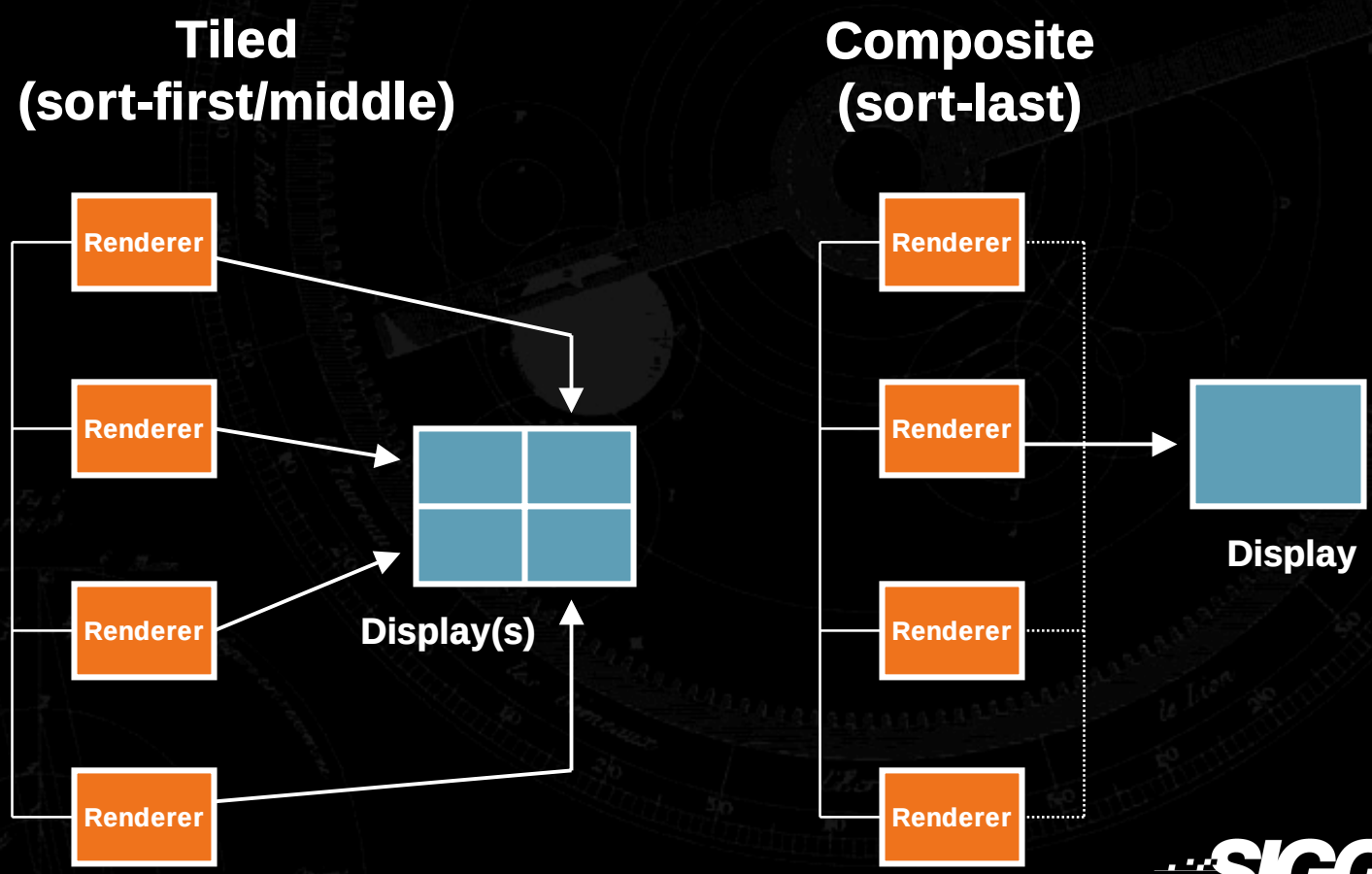
Rendering and Sorting



Polygon Rendering Pipe

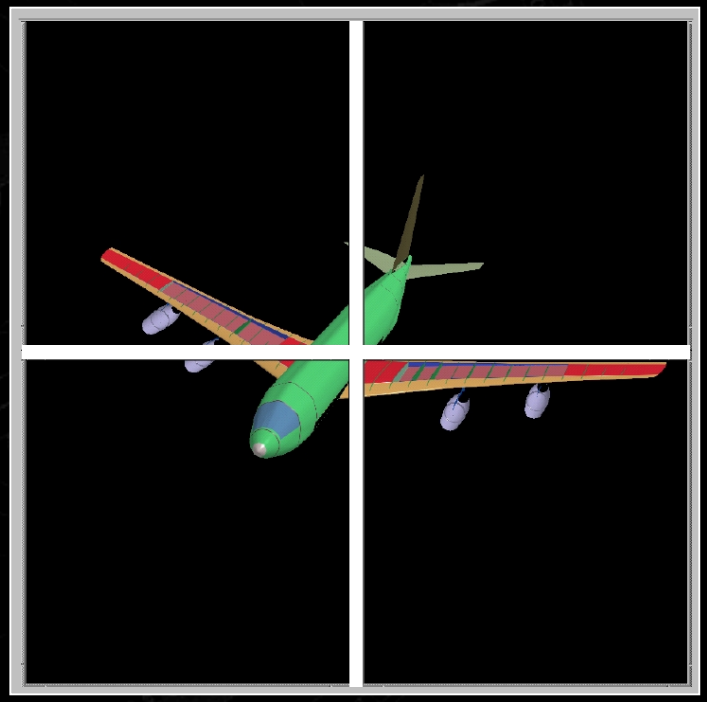


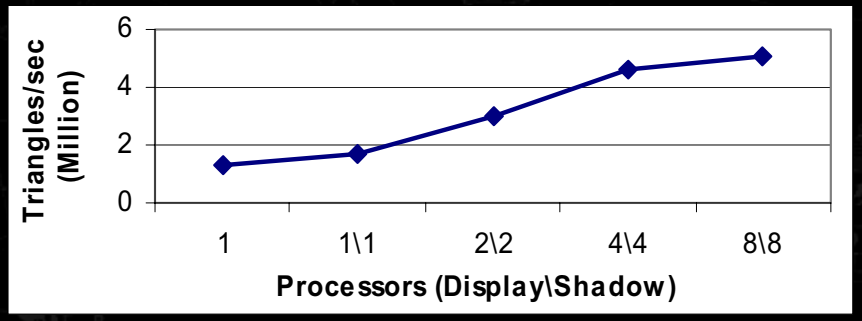
Tiled vs. Single / Composite Displays



Parallel sort-first rendering

- Useful for high-resolution, tiled display walls
- Higher frame rates possible
- Can leverage temporal coherency
- Tougher load balancing (data per tile can vary significantly)
- Communication traffic proportional to data size (data migrates to node(s) associated with each display tile)

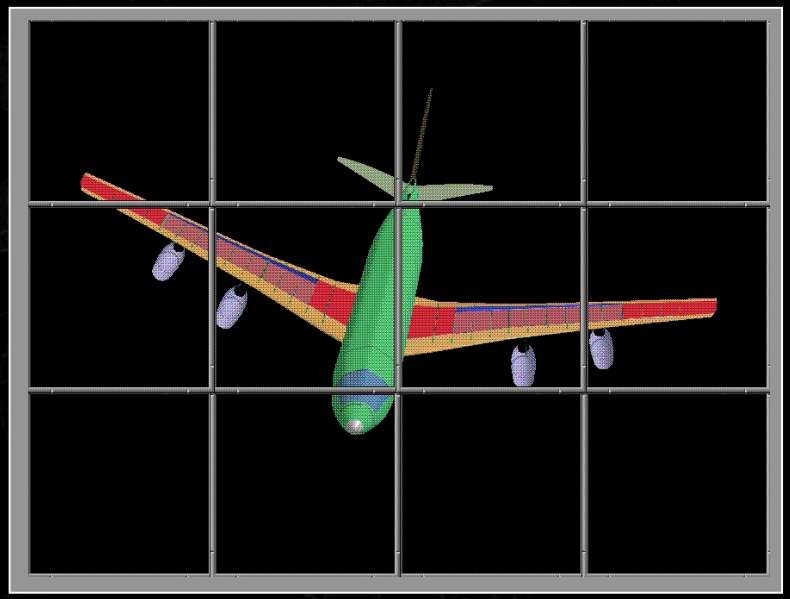




Early results: TDL performance on 1 million polygon dataset, Horizon cluster

Parallel sort-first rendering "TDL" library

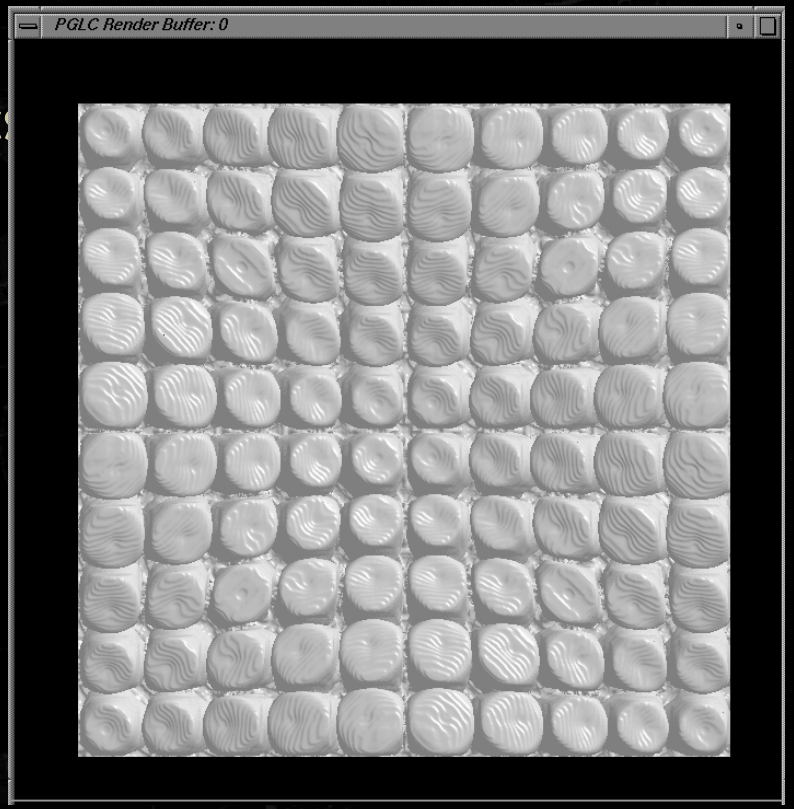
```
#include <tdl.h>
void main(int argc, char *argv[])
{
  TDL_Init(numXtiles,numYtiles,myX,myY);
  TDL_SetGeometry(coords,normals,colors,
                  triangles,numTriangles);
  while (1) {

    TDL_SetViewMatrix(vMatrix);
    TDL_GetGeometry(&coords,&normals,
      &colors, &triangles,&numTriangles);
       ...
    OpenGL Calls
  }
}
```

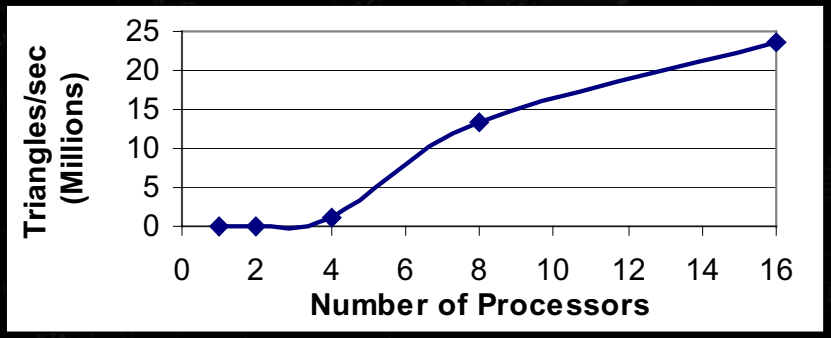


- Simple API; leaves rendering specifics up to the application
- Handles displays with overlaps or mullions
- Sort-first work giving way to WireGL & Chromium

Parallel sort-last rendering

- Back-end image compositing
- Good load balancing characteristics
- Performs well on large datasets
- Communication scales with image size (each node computes full resolution image)
- Low frame rates (compositing overhead)

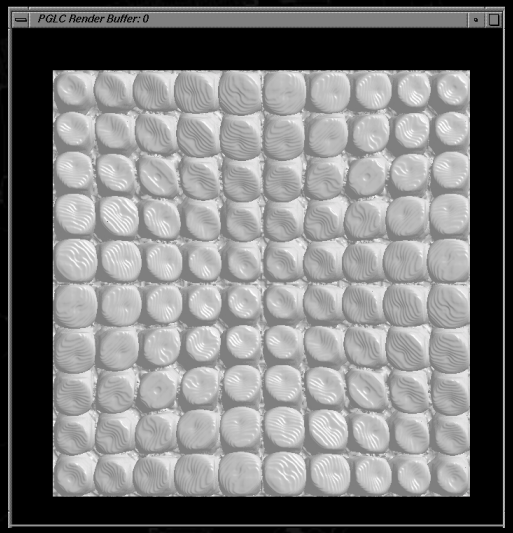




Early results: PGLC performance on 26 million polygon dataset, Horizon cluster

Parallel sort-last rendering "pglc" library

```
#include <pglc.h>
void main(int argc, char *argv[])
{
pglc_Init();
pglc_Wincreat(width,height,xPos,yPos,title);

while (1) {

      Computation
          .
      OpenGL Calls
          .
      frameBuffer = pglc_Flush(COMPRESSED_TREE);
   }
}
```



- Simple API; leaves rendering specifics up to the application
- Allows for implementation/use of different compositing schemes

ViCky - Sandia's Big Viz Cluster

RiCk - Render Cluster (cost ~$500K)

- 64 Compaq Professional Workstation SP750
- Pentium III Xeon 800 MHz processor
- nVidia GeForce graphics - Elsa ERAZOR² X 32 MB DDR
- 512 MB memory (32 GB total)
- 18 GB, 10 krpm local disks (1.2 TB total)
- Will drive a 16Mpixel tiled display wall

Feynman - Data Cluster

- 72 Compaq Proliant 1850R computers (older)
- Dual Pentium II 400-MHz processors
- 512 MB memory (37 GB total)
- 4x9.2 GB, 10 krpm local disks (2.6 TB total)
  - + some network RAID
- >3.7 GB/s bandwidth (disk to memory)

Upgrading with ~44 I/O & Communication Nodes and ~20TB disk (multiple fibre-channel RAIDS)

Dual boot - Windows 2000 and Linux

Servernet-2 interconnect (changing to Myrinet 2000)

Gig Ethernet / 100-MHz Ethernet external connects

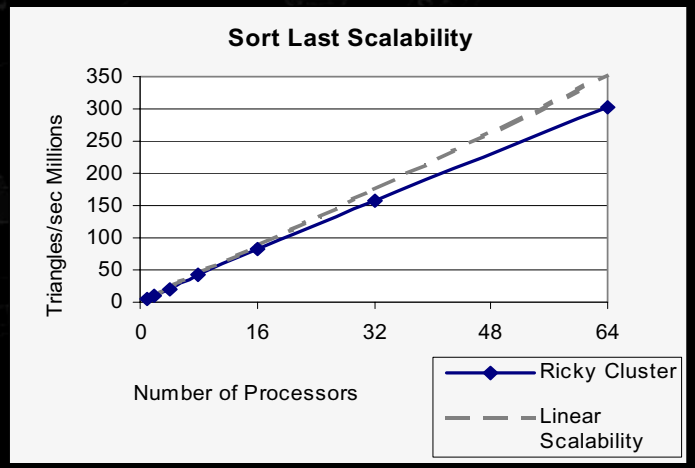
SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

PGLC results on the big cluster

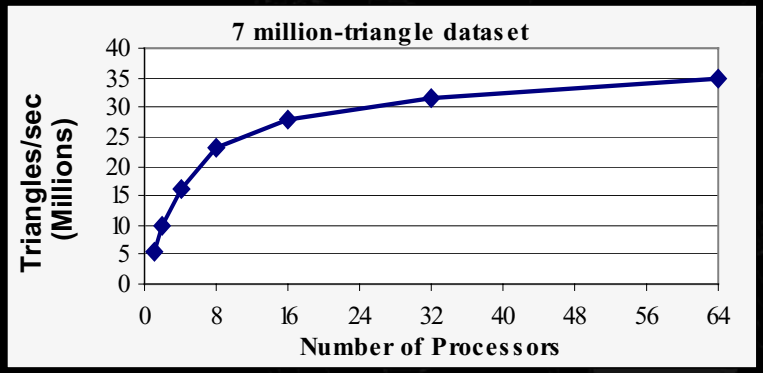
*Demonstrated 300 Million triangles per second**

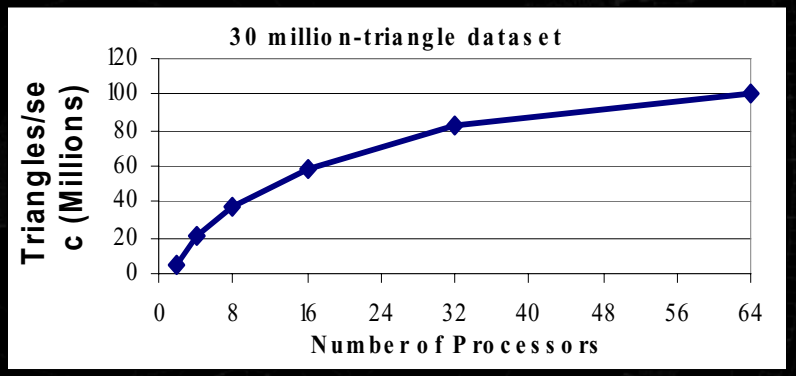
- **470 Million triangles in less than 2 sec per frame**
- **>100 times the performance of SGI IR pipe (for our applications)**
- **Using 64 ViCky rendering nodes**
- **Sort last, 1024x780 images**
- **Windows 2000 OS**

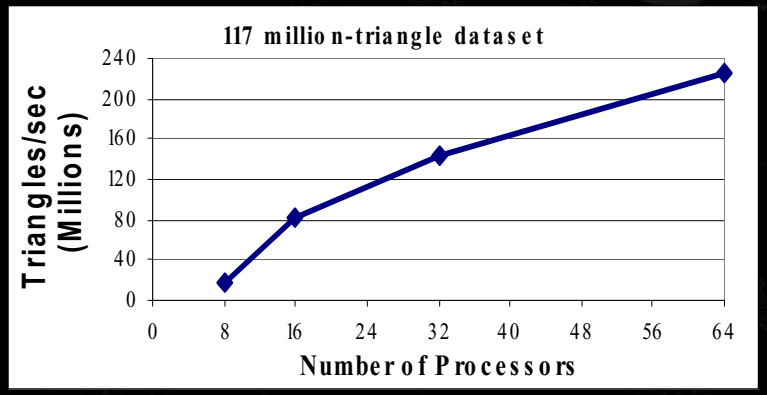
*Some issues with image correctness and the 470M-triangle data.

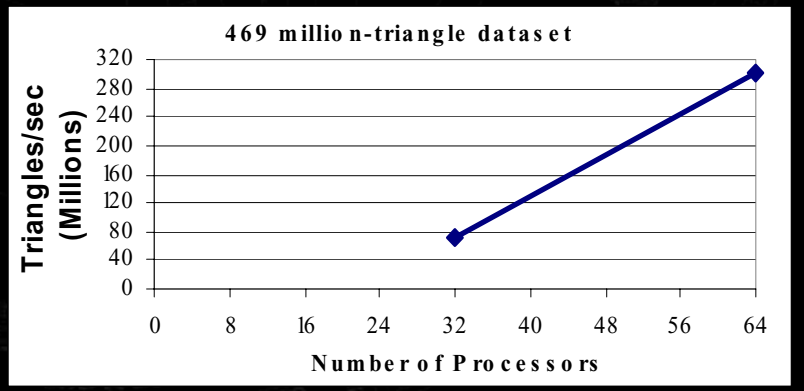


More PGLC results



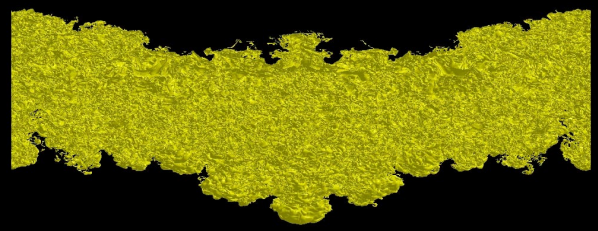


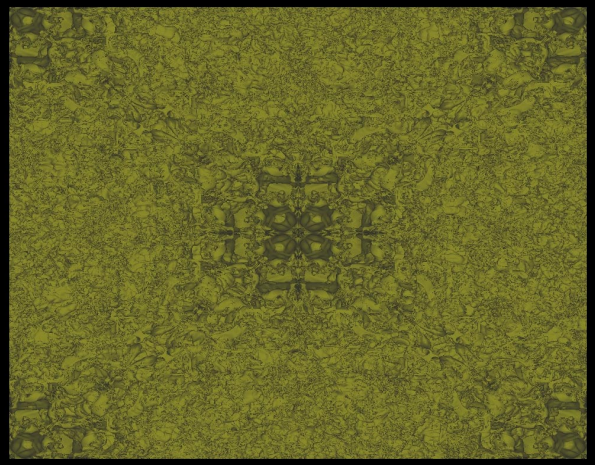


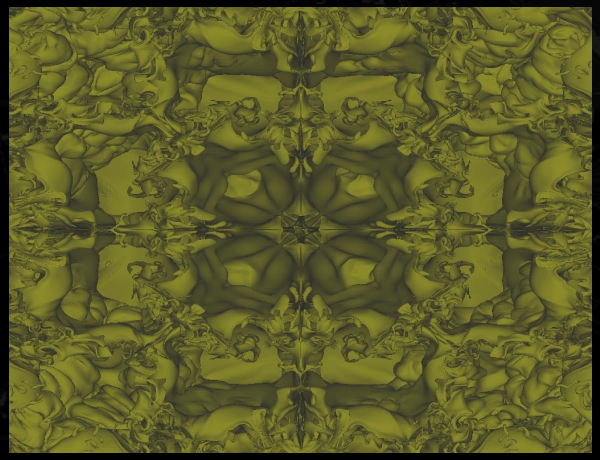


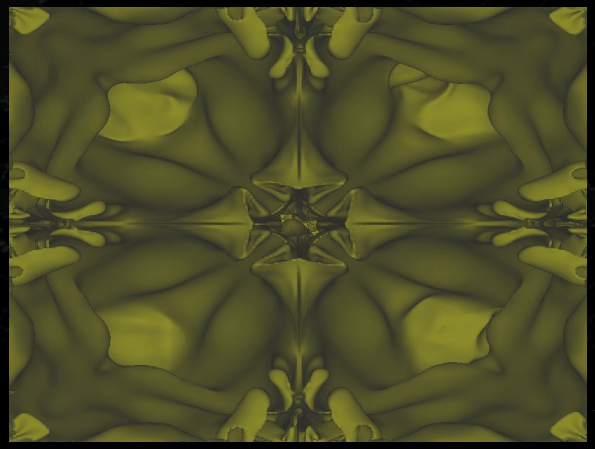
SIGGRAPH 2001
EXPLORE INTERACTION AND DIGITAL IMAGES

470-Million Triangle Data









SC99 Gordon Bell PPM dataset - Art Marin et al, LLNL

Isosurface – Dan Schikore, CEI (previously LLNL)

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

Optimization Tidbits

Frame Read-back …use optimal format for OpenGL glReadPixels()
[observed performance differences as much as 30X]

- Color GL_BGRA GL_UNSIGNED_BYTE
- 24-Depth GL_DEPTH_COMPONENT GL_UNSIGNED_INT
- 16-Depth GL_DEPTH_COMPONENT GL_UNSIGNED_SHORT

Use of run-length-encoded/active-pixel-encoded images

- RGBA 4-tuples treated as single 4-byte quantity
- 10-15X reduction in early composition stages

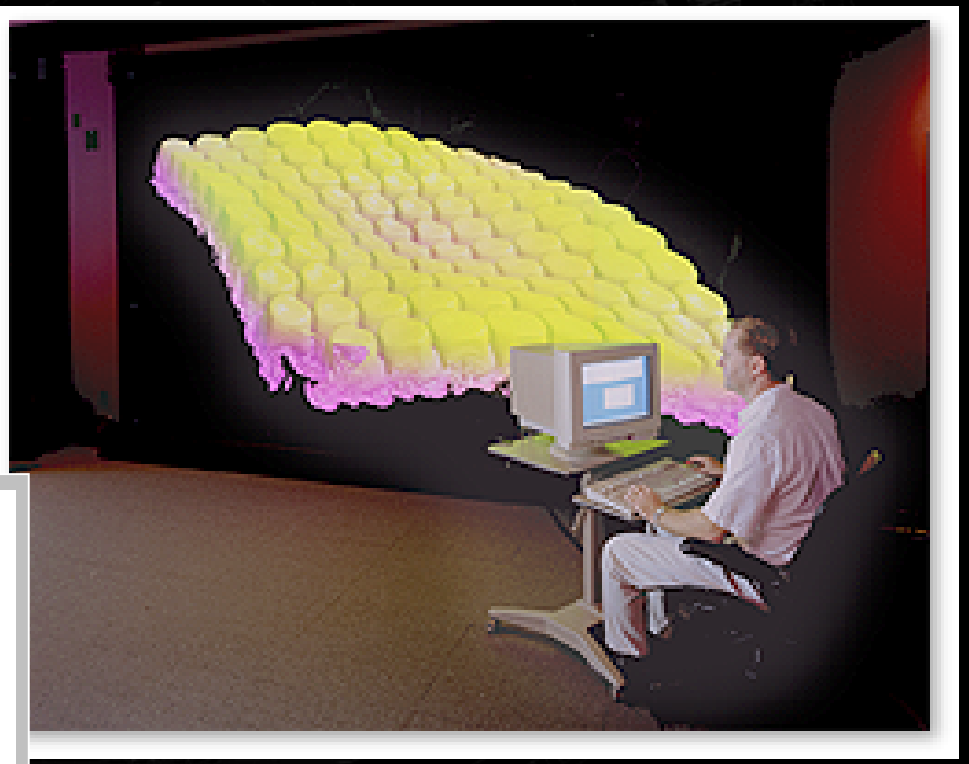
Composite compressed data directly

- Save compression/decompression at each step

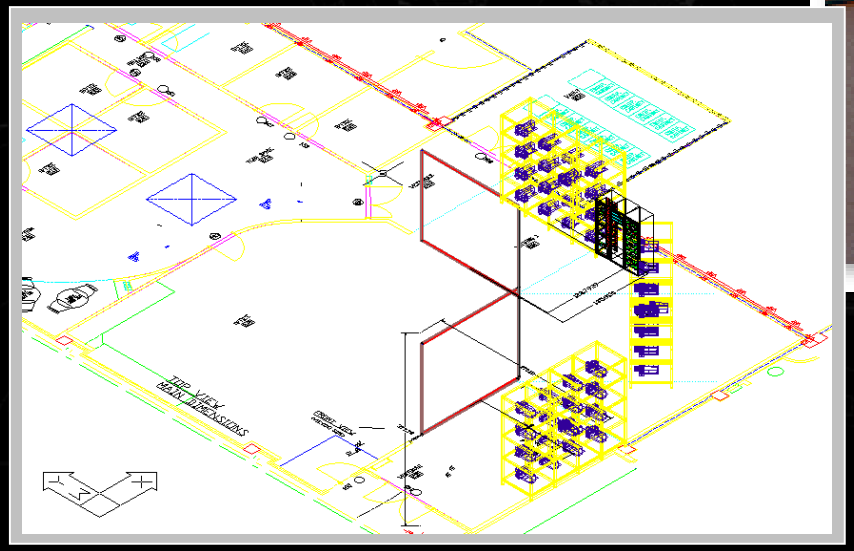
Observed TOTAL overhead reduction ~64% (500ms to 180ms for 64 nodes at image resolution of 1024x768)

We are working on sort-last approaches for tiled-displays

Sandia/California's Power Wall
4x3 tile-display
Approximately 16 Mpixels

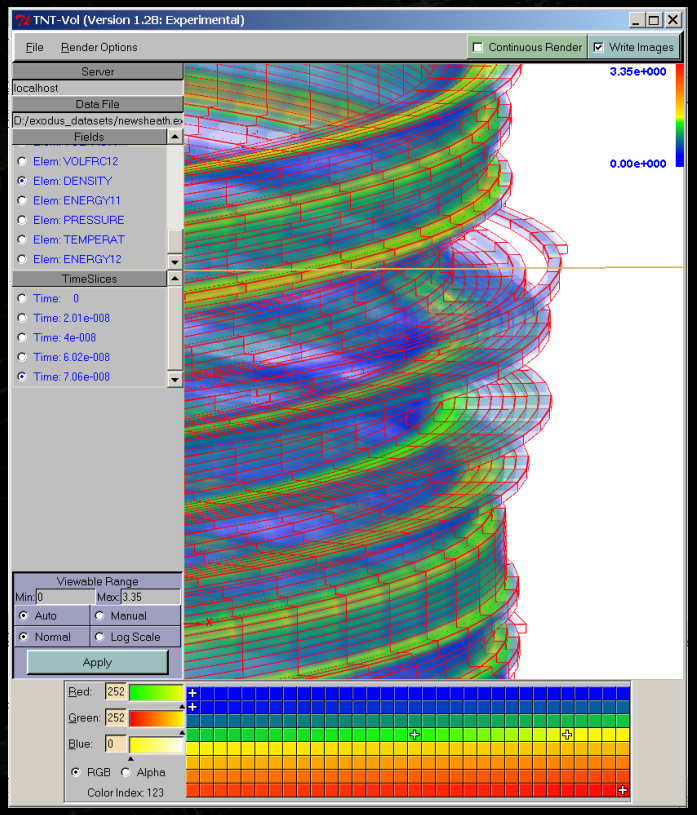


Sandia/New Mexico Facility
Incorporates 16-tile display
Planned for 48-tile, 3-screens



Volume Rendering

- Projection-based: Uses graphics hardware
- Cells sorted and projected onto image in order from back to front, with blending
- High frame rates possible
- Client/Server architecture
  - Desktop delivery
  - Server easily swapped out
- Approximate sorting for best performance; integration of "Z-sweep" or "BSP-XMPVO" for exact sort
- Multi-display load balancing issues
- Early results: 350,000 unstructured hexahedral cells per second



TNTVol on 700,000 hexahedral cells

More Clusters

Sandia/NM testbed cluster (cost ~$150K)

- **16-node Dell-620/nVidia+Matrox cluster, Myrinet or Servernet2**

Sandia/CA cluster (cost ~150K)

- **16-node Dell-620/nVidia cluster, GigE/Myrinet**

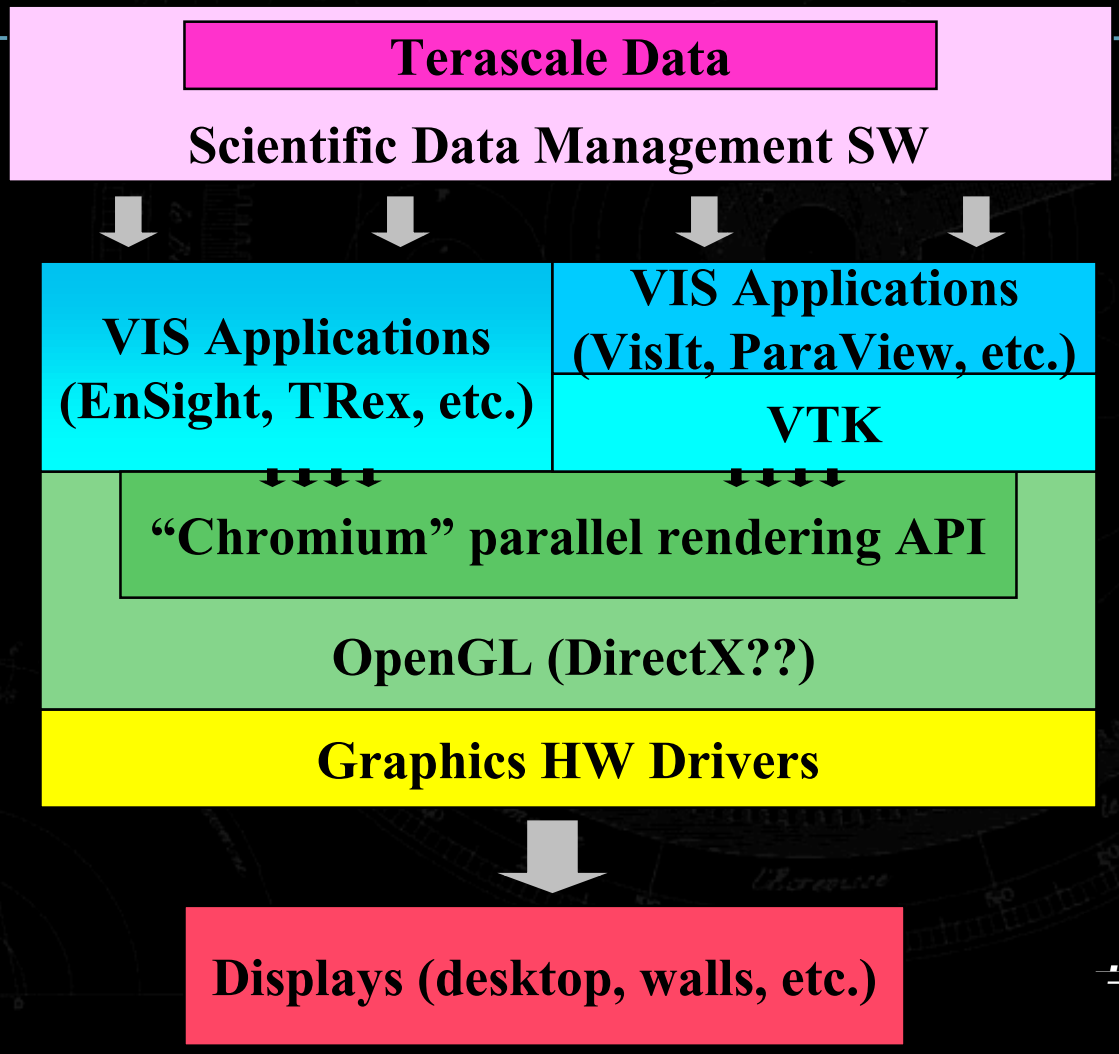
New ~64-node cluster ("Europa")

- **One each for classified/unclassified (with "RiCk")**

VIEWS Partnerships

- Academic partnerships (Stanford, Princeton, …)
- CEI -- under ASCI/VIEWS tri-lab contract to parallelize EnSight product, including for cluster-based graphics architectures
- Kitware -- under ASCI/VIEWS tri-lab contract to develop parallel/distributed VTK that will run on cluster-based graphics architectures
- Parallel, Distributed OpenGL Rendering API/Engine effort
- Linux graphics drivers (nVidia, Precision Insight)
- Scalable Visualization RFP

Scalable Visualization Software



Los Alamos Cluster Visualization

Allen McPherson
Los Alamos National Laboratory

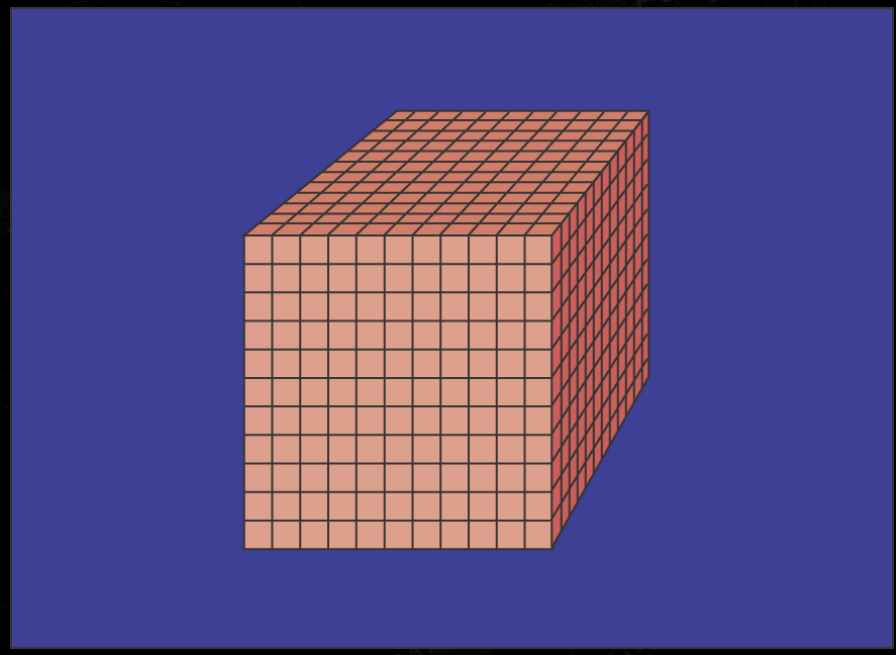
Agenda

Volume rendering overview
Cluster-based volume rendering algorithm
Back-of-the-envelope analysis
Cluster architecture
Software environment
Recent results
Future work

What is Volumetric Data?

3-D grid or mesh
Data sampled on grid
Samples called "voxels"
Many grid topologies

- Structured
  - E.g. rectilinear
- Unstructured



How is Volume Data Generated?

Sensors

- CT scanners
- MRI

Simulations

- Fluid dynamics

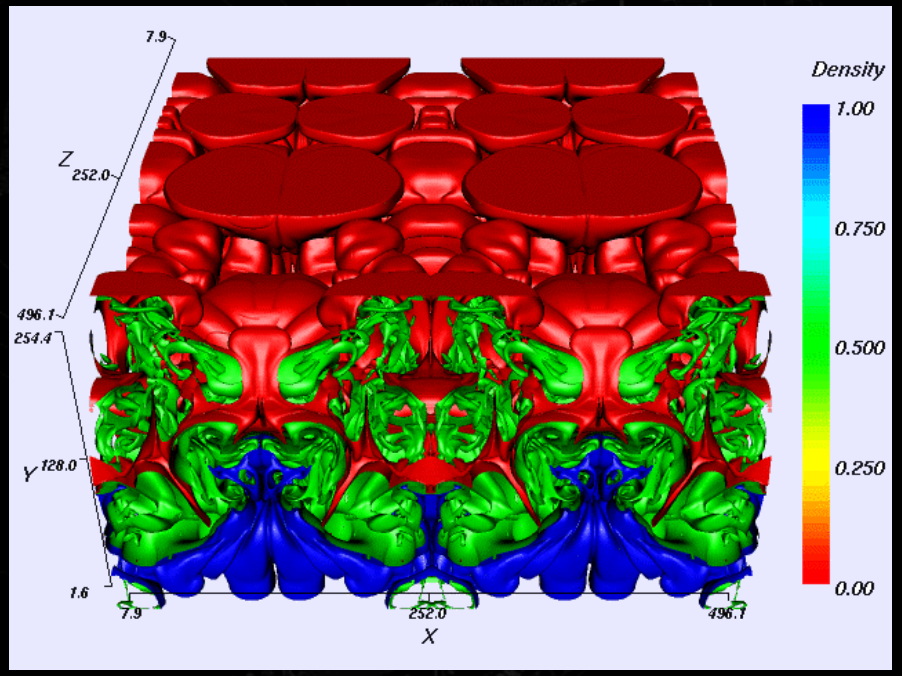
Measured data

- Ocean buoys

Looking at Volumetric Data

Constant value surface

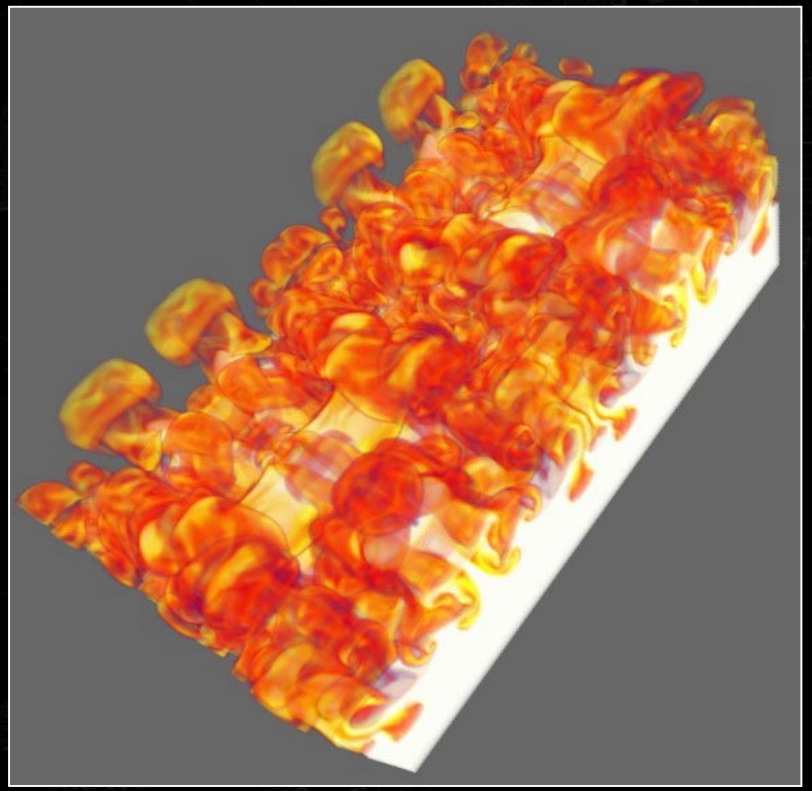
- Isosurface algorithm
- Polygonal data generated
- Don’t see entire volume
- Polygons usually generated in software
- Polygons rendered with hardware



Looking at Volumetric Data

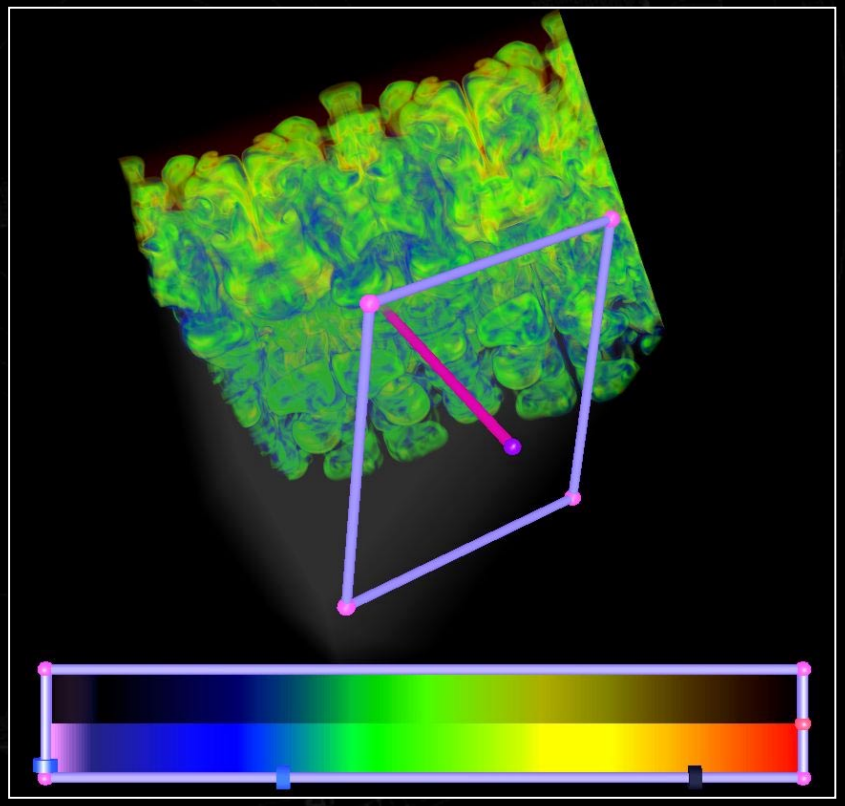
True volume rendering

- Treat field as semi-transparent medium
- "blob of Jello"
- Can see entire volume



Transfer Functions

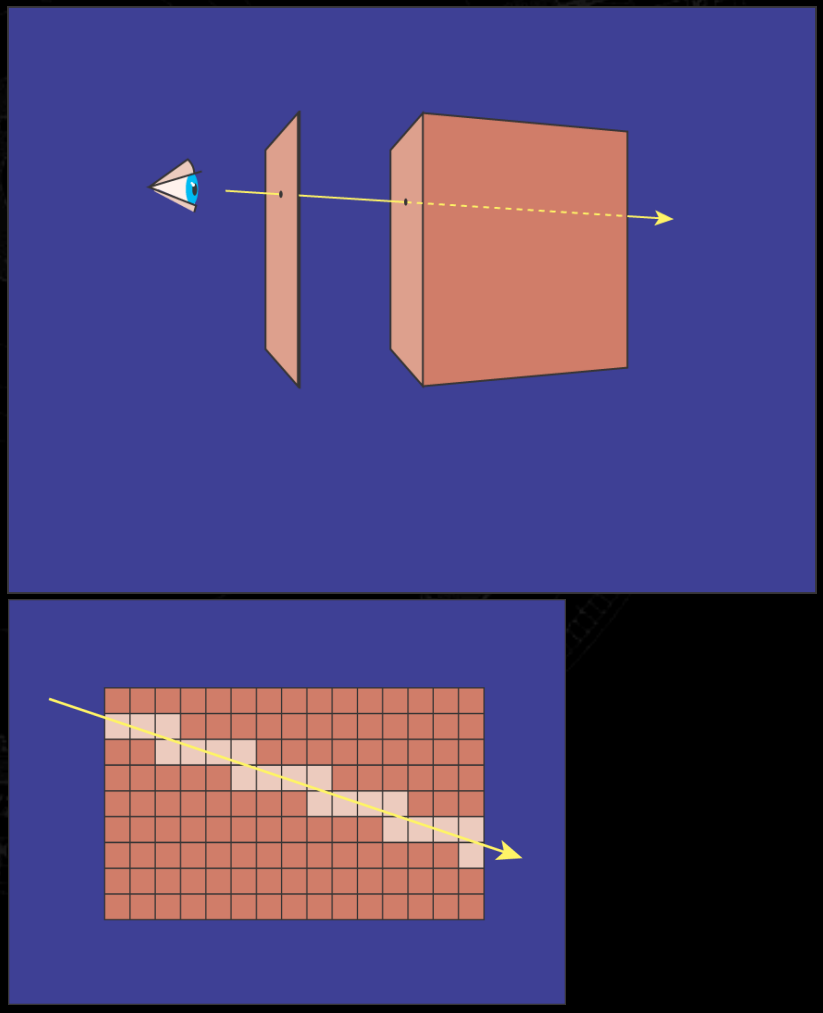
- Indirectly maps data to color and opacity
- Allows user to interactively explore volume



Software Volume Rendering

Ray casting

- Image order algorithm
- Trace ray through image plane and into volume
- Sample volume at regular intervals along ray
- Combined samples yield ray's pixel value (compositing)



Hardware Volume Rendering

Software approaches are too slow

- Interactivity required for exploration

Use texture mapping hardware to accelerate

- Textures emulate the volumetric data
- Hardware lookup tables accelerate transfer function updates

Use parallelism for large volumes (multiple hardware pipes)

Texture Mapping Approach

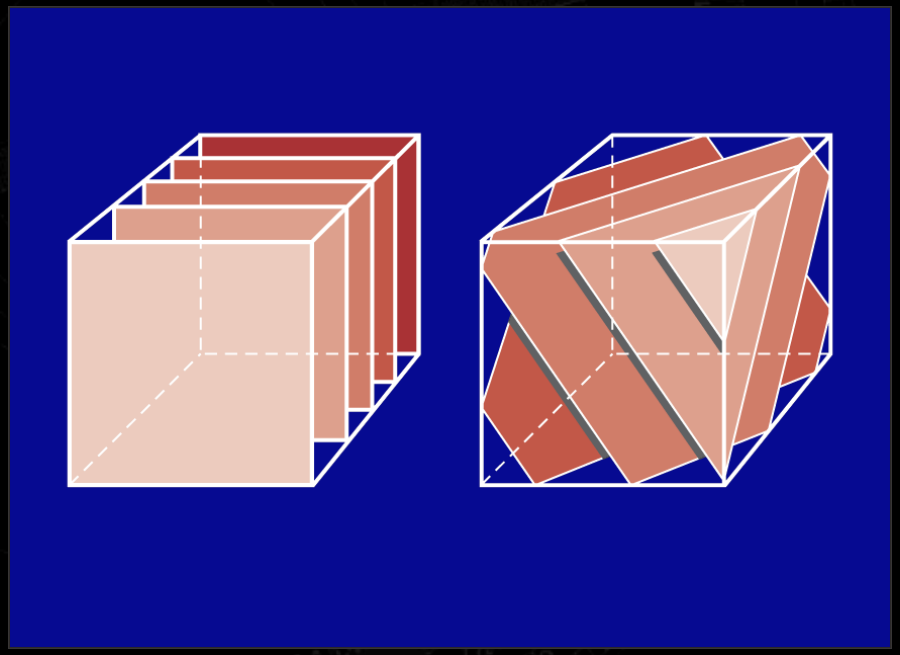
Texture is volume

- 3-D texture
- Many 2-D textures

"Cleave" 3-D volume with slice planes

Composite resultant images in order

Essentially parallel ray casting



Early Experience at Los Alamos

- Problem: visualize large volumetric data (1024^3) interactively
- Use texture-based approach for speed
- Single pipe can't handle large volumes
- Use multiple pipes in combination to render large volumes

Large SGI-based Solution

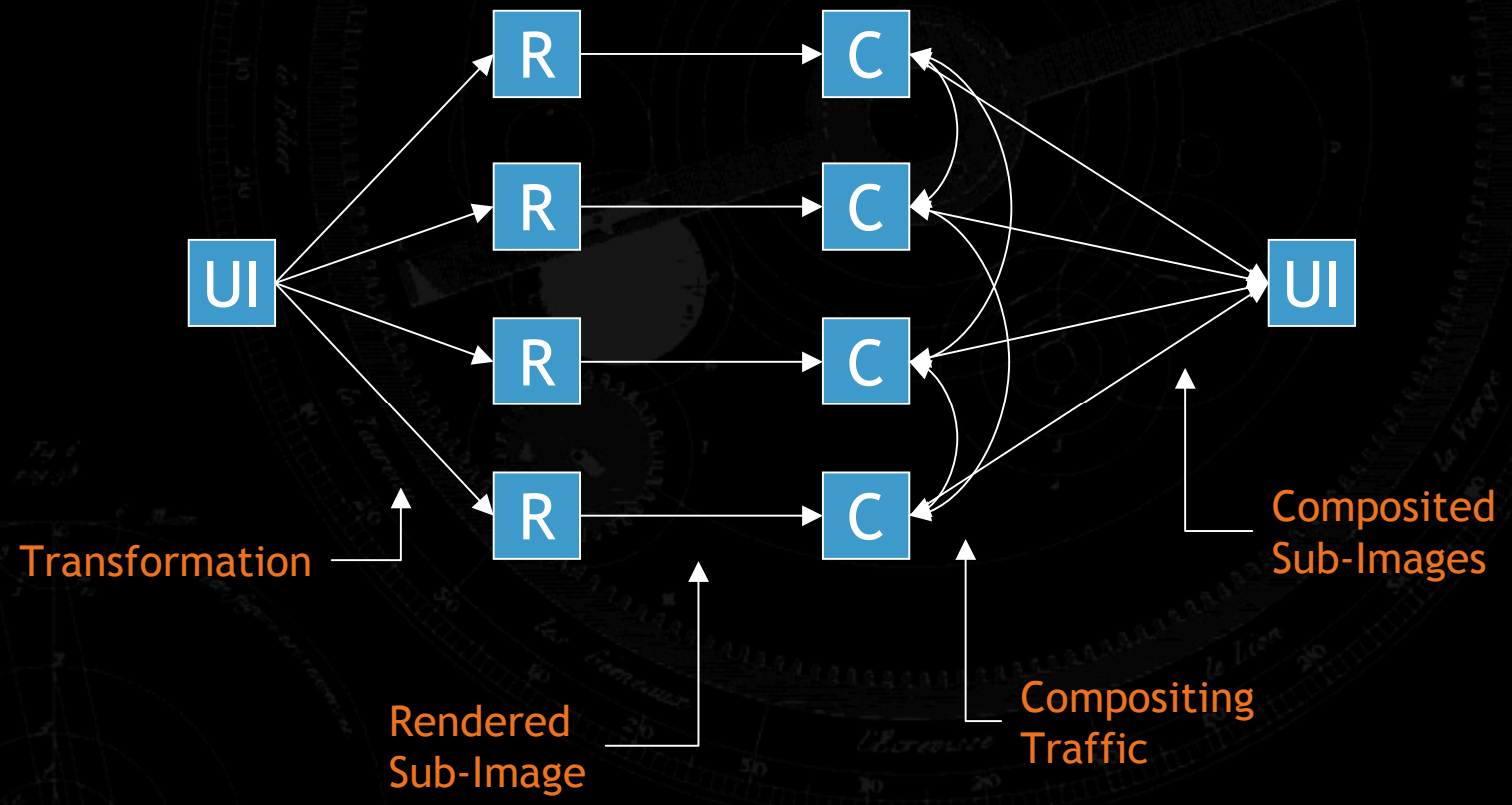
- 128 processor Onyx 2000
- 16 Infinite Reality graphics pipes
- 1 Gvoxel volume rendered at 5 Hz
- Want to accomplish the same goal (or better) using less expensive, commodity-based, solution
- Our volumes will get bigger—8K³

Cluster-based Solution

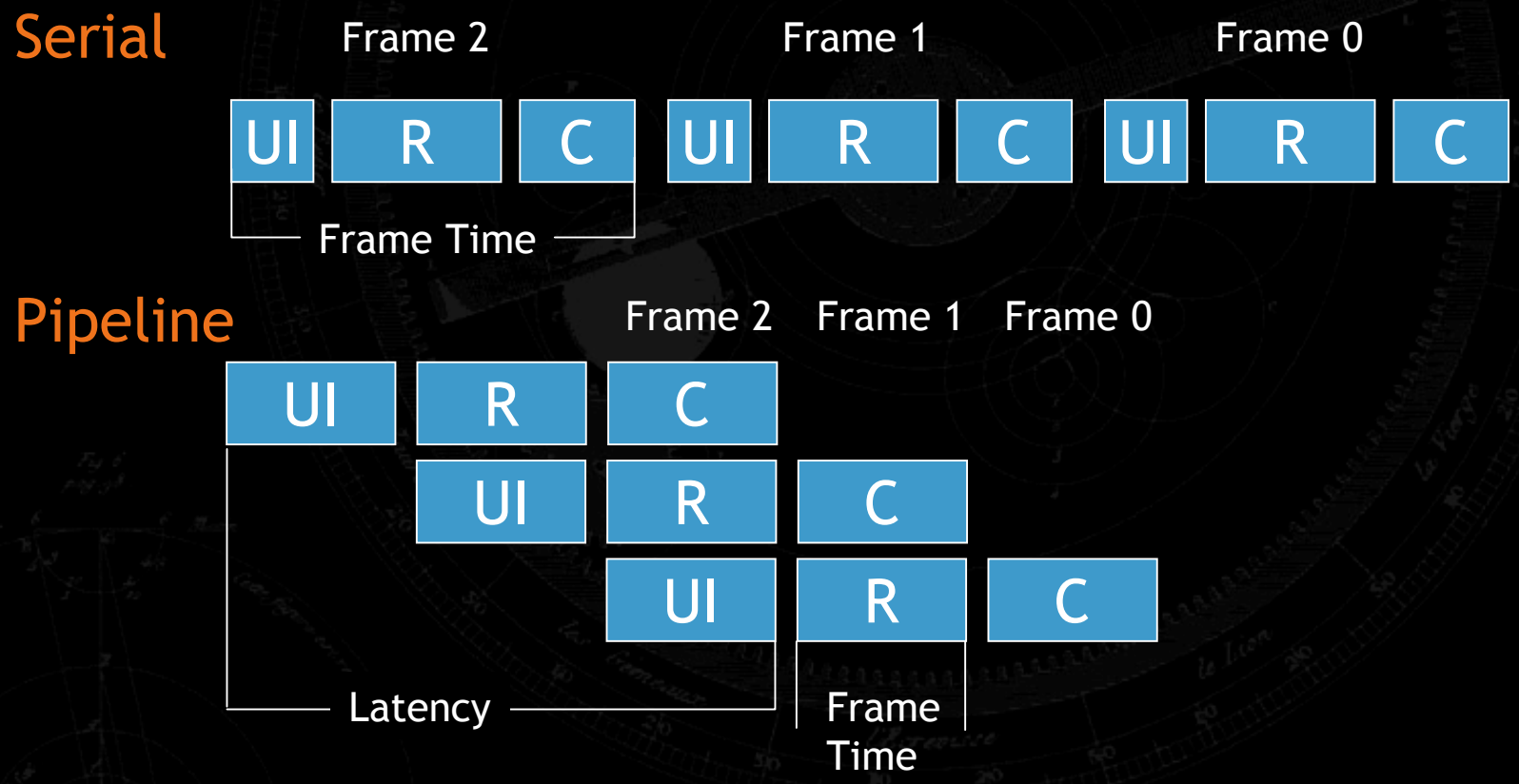
Algorithm similar to large SGI solution

- Break volume into smaller sub-volumes
- Use many PC nodes with commodity graphics cards to render sub-volumes
- Read resultant images back and composite in software using interconnected cluster nodes
- Organize as pipeline for speed

Algorithm Schematic



Serial vs. Pipelined



Pipeline Issues

Frame time = time of longest stage

- Need to **balance** stage times

Deep pipelines can induce long latency

- Keep pipelines short

"Circularity" of pipeline is troublesome

- Communications programming is tricky

Back-of-the-Envelope

Analyze feasibility

- Examine “speeds and feeds” of each component
- Test against theoretical numbers wherever possible
- Won’t guarantee success, but gets us in the ballpark

Cluster Components

Initial hardware selections

- CPU: dual Intel
  - want commodity PC
- Graphics: Intense 4210
  - using 3-D texture
- Network: GIG-E
  - Fast commodity network
  - Reusable at completion of project

Bounding Parameters

Graphics card texture memory

- Dictates size of volume that can be rendered

Graphics card fill rate

- Dictates speed of actual volume rendering

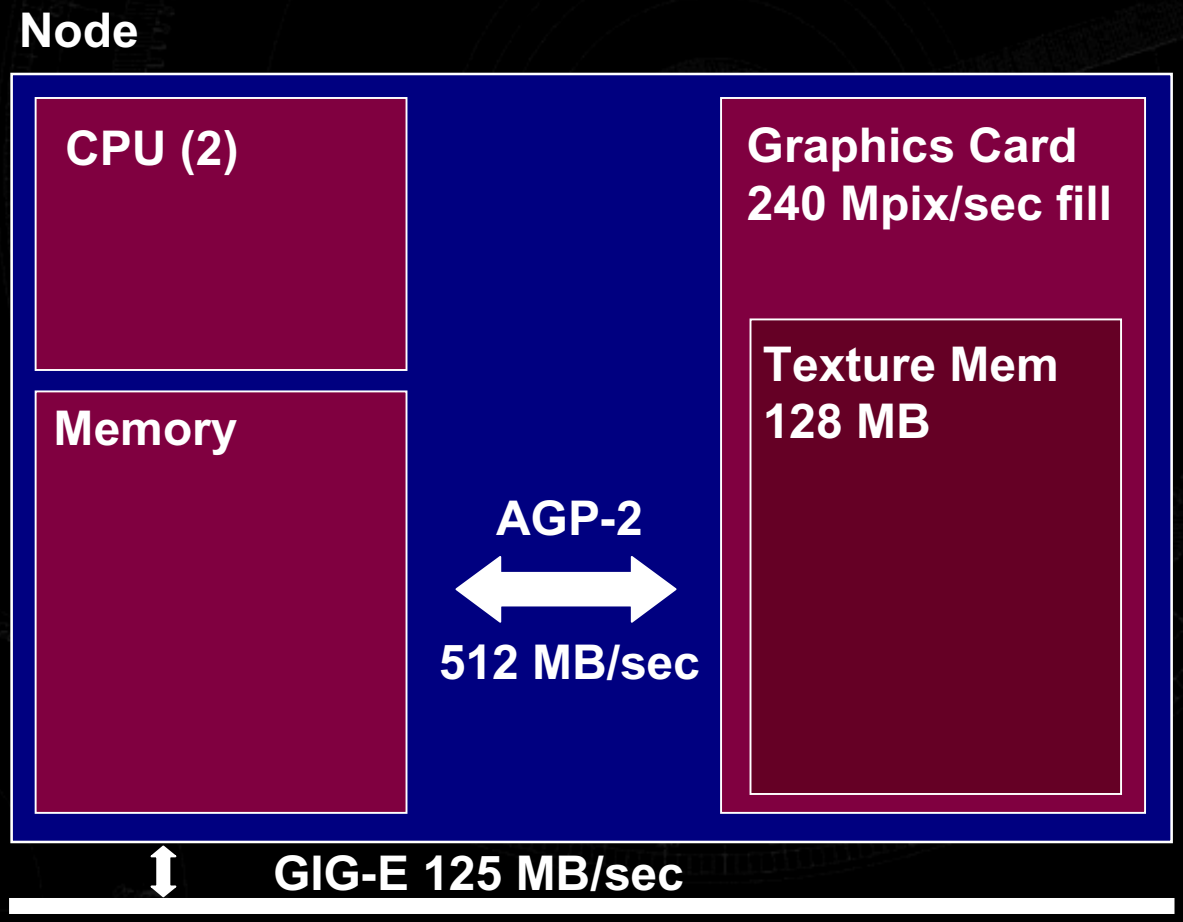
Framebuffer readback rate

- How fast rendered sub-frame can be read to host

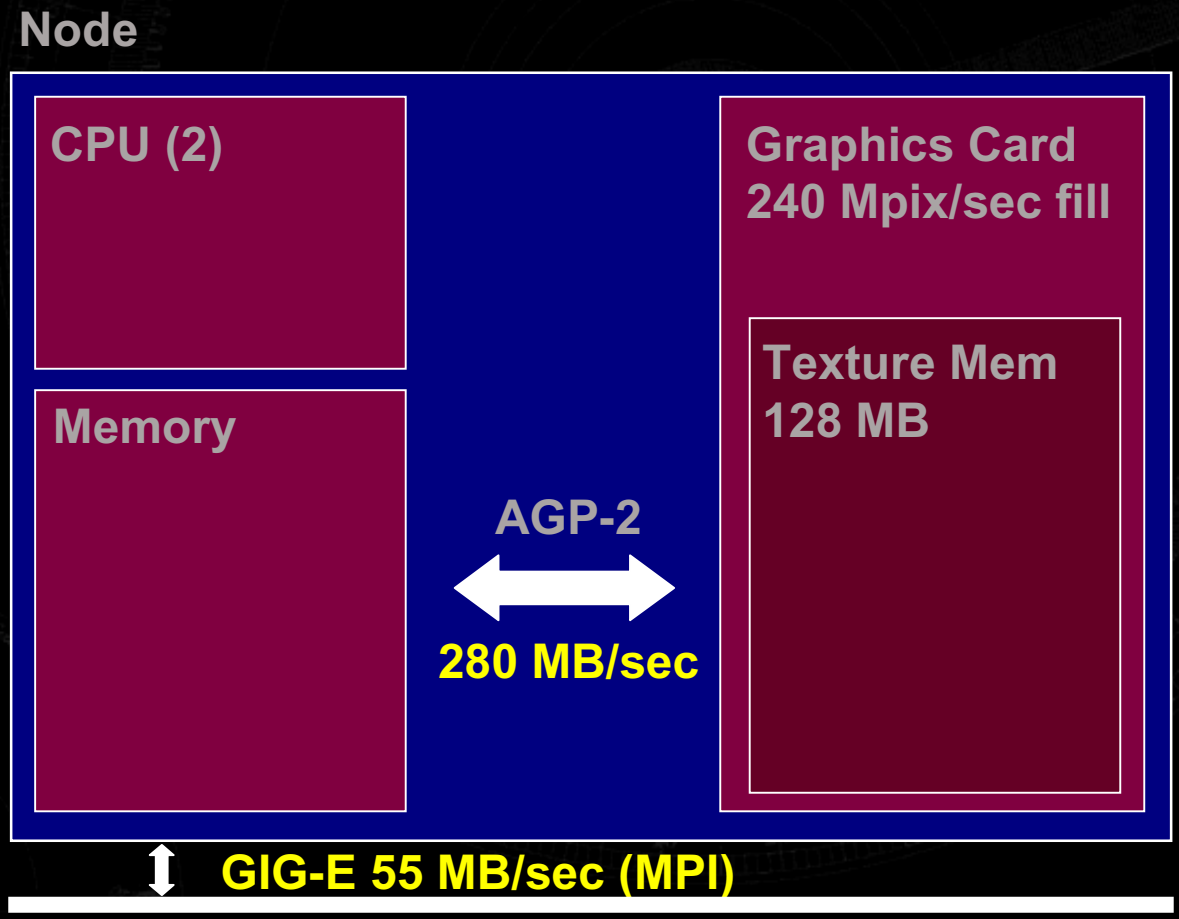
Network speed

- How fast images can be moved through the cluster

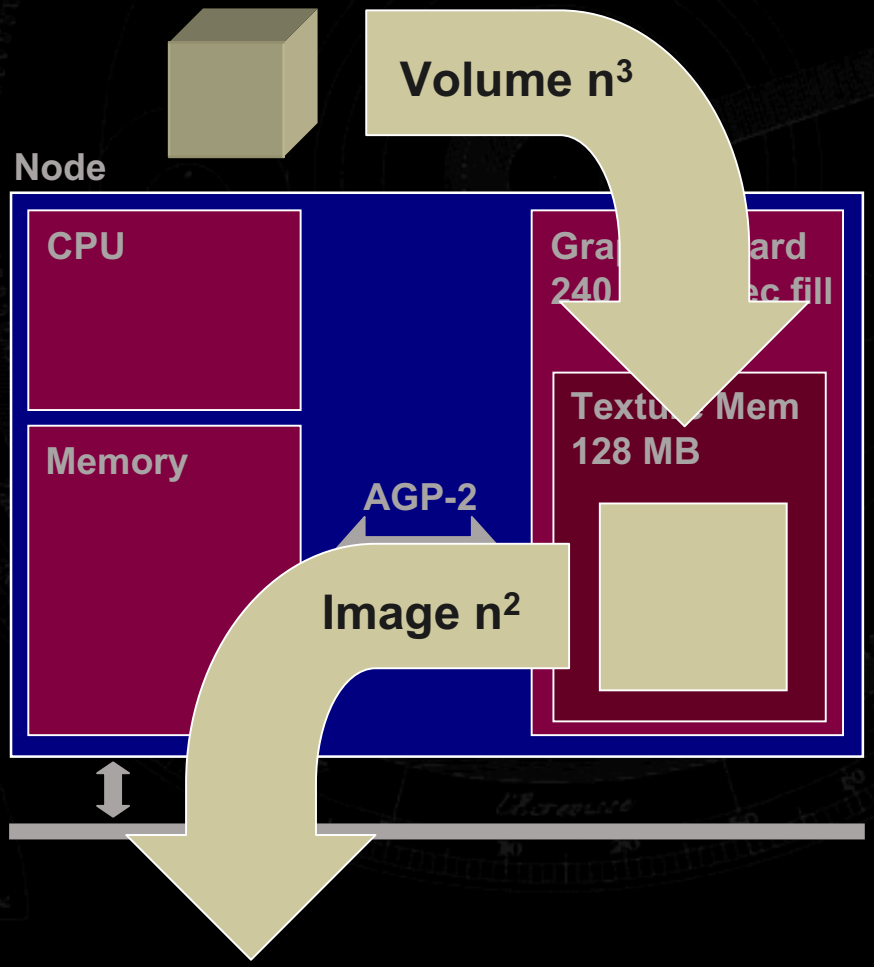
Bounding Parameters (theory)



Bounding Parameters (tested)



Data Magnitude



Limit 1: Rendering

240 Mtex/sec

- At 5 FPS budget ~50 Mtex/frame
- 1-1 pixel-voxel gives 50 Mvoxel volume
  - ~512x512x256 (64 MB through TLUT)
  - 32 nodes gives 2 Gvoxel volume
- Theoretical number
  - Conservatively use ½ of theoretical
  - Back to 1 Gvoxel volume

Limit 2: Image Readback

280 MB/sec AGP-2 tested

- Assume that we render into a 1024^2 image
  - Matches volume resolution to screen resolution
- RGBA gives 4 MB/frame
- 280/4 = 70 FPS
  - Well within budget

Limit 3: Network Performance

55 MB/sec tested on GIG-E with MPI

- 4 MB (or smaller) images
- 55/4 = 11 FPS
- Within budget, but…
  - May need to transport image multiple times per frame (render, composite, display)
  - 5 FPS allows only two image moves—may not be fast enough

Limit 4: Volume Download

Only required for time-variant data

- 64 MB volume from Limit 1
- At 5 FPS requires 320 MB/sec download
- Tested AGP-2 limits to 280 MB/sec
- Would need matching I/O
  - 320 x 32 nodes: 10 GB/sec aggregate I/O

Balanced Pipeline Stages?

UI

- Very fast, small data transfers (transform, TLUT)

Render

- 200 ms/frame + 4 MB image transfer

Composite

- Composite operations + 4 MB image transfer

Pipeline forces equal stage lengths

Network time need to be considered

Los Alamos KoolAid Cluster

Cluster Compute Hardware

36 Compaq 750

- Shared rendering/compositing nodes
- 4 nodes used for UI and development
- Dual 800 MHz Xeon
- 1 GB RDRAM per node
- Intel Pro-1000 GIG-E card

Cluster Compute Issues

Intel 840 chipset allows simultaneous:

- AGP transfers
- Network transfers
- CPU/memory interaction

Some problems with chipset

- Poor PCI performance when compared to Serverworks—slows networking

Cluster Network Hardware

Extreme GIG-E switch

- Supports jumbo packets
- Full speed backplane
  - Simultaneous point-to-point transfers
- Intel Pro-1000 GIG-E cards
  - Tested for this application

Cluster Network Issues

GIG-E is relatively slow and inefficient

- Protocol processing eats CPU
- Extreme switch is expensive, but nice

Need to test actual communications patterns

- Simple "netperf" style is not enough
- Test with communications library to be used (MPI)

Numerous driver issues—test, test, test!

All GIG-E equipment is re-usable

Cluster Graphics Hardware

3Dlabs Wildcat 4210

- 128 MB texture memory
- 128 MB framebuffer memory
- 3-D texture hardware

Cluster Graphics Issues

Sub-optimal compared to recent alternatives

- Poor fill rate
- AGP-2 interface
- Expensive: ~$4000/card
- Lacks nifty new features (DX8, etc.)

Can clearly do better next time

Software Environment (OS)

Windows 2000

- Not a religious issue with us
- Only OS with driver support for Wildcat 4210
- Best bet for drivers (commodity cards)
- Most application code portable to Linux
- Can experiment with DX8 features later

Software Environment (Rendering)

OpenGL

- 3-D textures for volume rendering
  - Not in pre-DX8 versions from Microsoft
- Solid support on Wildcat 4210

Softwarecompositing

- Have CPUs with nothing to do
- Completely general for future experimentation

Software Environment (Networking)

MPI

- Argonne MPICH implementation
- Easy to learn and use
- Implementation adds opaque layer which makes troubleshooting difficult
- A few Win2K issues
  - General lack of tools (e.g. log viewing)
  - Tag limit of 99 (MS licensing??)

Results

To be presented at Siggraph 2001

- See www.acl.lanl.gov/viz/cluster for latest

Future Work

Clusters of task-specific mini-clusters

- Rendering, compositing, I/O, display
- Possibly specialized interconnect between clusters
  - DVI
  - Fiber Channel
- Optimal interconnect for individual mini-clusters
  - Myrinet-2000
  - Simple 100 Mb Ethernet

Future Work (Rendering Cluster)

Take rendering cluster to 64 nodes

- Still Compaq 750s
- New nVidia/ATI cards when 3-D texture-capable
- May use Microsoft DirectX 8 vs. OpenGL
- Doesn't need high speed interconnect
  - Just transforms and TLUTs
  - Does need high speed connection to compositing cluster

Future Work (Compositing Cluster)

64 1U compositingnodes

- Dell PowerEdge 1550
  - Single 1 GHz PIII
  - Serverworks chipset
- Interconnected with Myrinet-2000
  - 2 Gb/sec interconnect
  - Much faster than GIG-E, much less CPU overhead
- May run Linux
  - No need for Win2K since no graphics cards

Acknowledgements

John Patchett
Pat McCormick
Jim Ahrens
Richard Strelitz

Stanford's MultiGraphics: Scalable Graphics using Commodity Components

Greg Humphreys

Stanford University

SIGGRAPH 2001
EXPLORE INTERACTION AND DIGITAL IMAGES

Overview

Chromium: Stanford/DOE visualization cluster

WireGL: Software for cluster rendering

- Application transparent support for tiled displays
- Parallel interface for scalable performance

Lightning-2: Image composition network

Interactive Room

- Mural
- Table

MultiGraphics Goals

Provide scalable graphics on commodity parts

- Processors
- Graphics Accelerators
- Networks
- Displays
- Other (image compressors, input devices, APIs)

Software support for rendering on clusters

- "Transparent" tiled displays
- "Obvious" parallel applications

Hardware support for rendering on clusters

- Image composition

Chromium Cluster

32 nodes, *each with graphics*

Compaq SP750

- Dual 800 MHz PIII Xeon
- i840 logic
- 256 MB memory
- 18 GB disk
- 64-bit 66 MHz PCI
- AGP-4x

Graphics

- NVIDIA Quadro2 Pro with DVI

Network

- Myrinet (LANai 7 ~ 100 MB/sec)

Cluster Rendering Software Goals

Remote rendering: as efficient as possible

- Well designed network protocol

Transparent support for tiled displays

- Sort-first distribution of graphics commands
- Efficient state management

Drop-in OpenGL replacement

- Support existing applications without modification
- Familiar immediate-mode API

Parallel interface for scalable rendering rates

- Controlled order of execution

Issues

Flexibility

- Heterogeneous computers/graphics/operating systems
- Continual upgrades of graphics and networks
- Ratio of components

Programming interface

- Scene graph or retained mode
- Immediate mode (time-varying)

  Parallel OpenGL

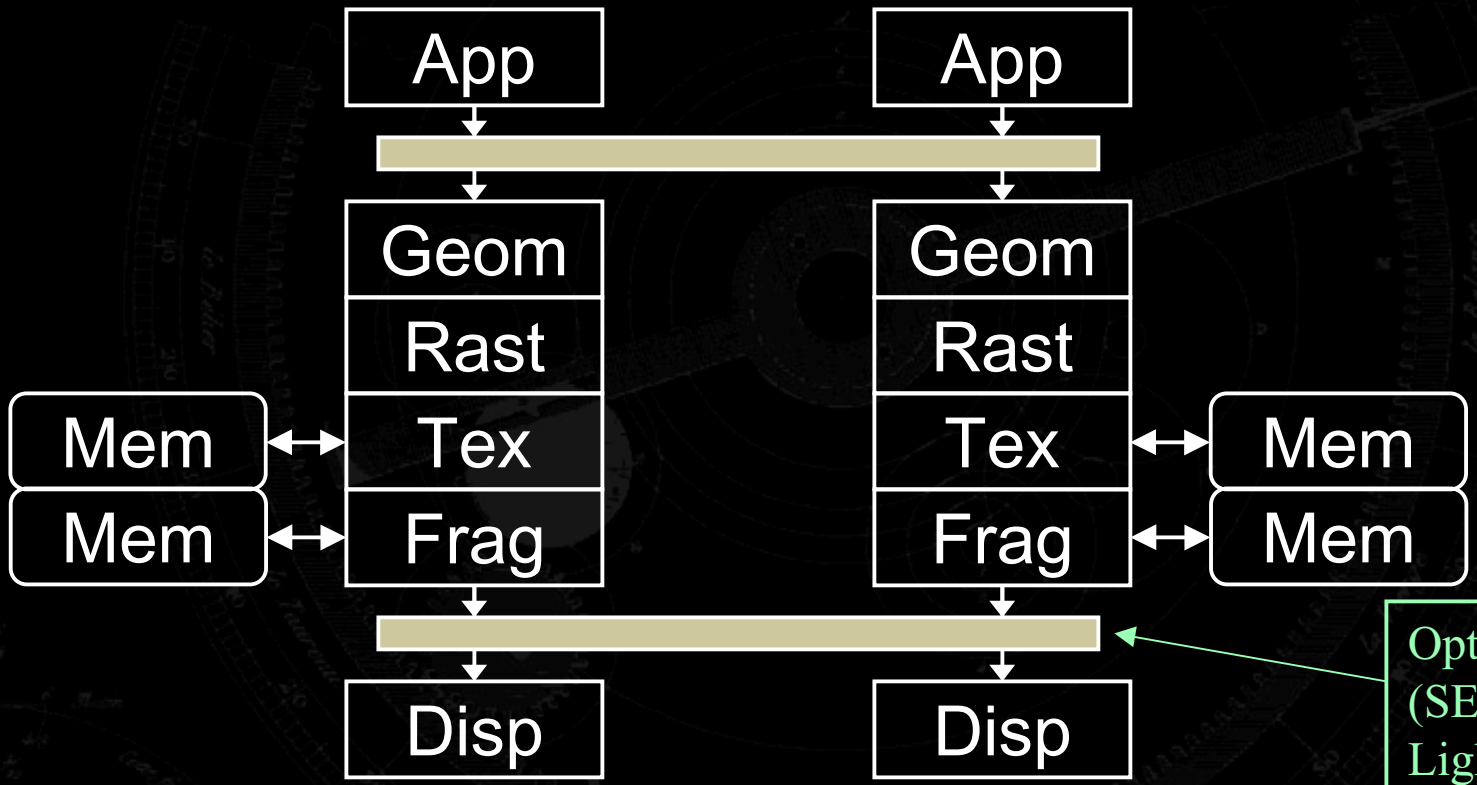
Communication and load balancing

- Granularity and sorting primitives

  Sort-first tiled

- Network substrate

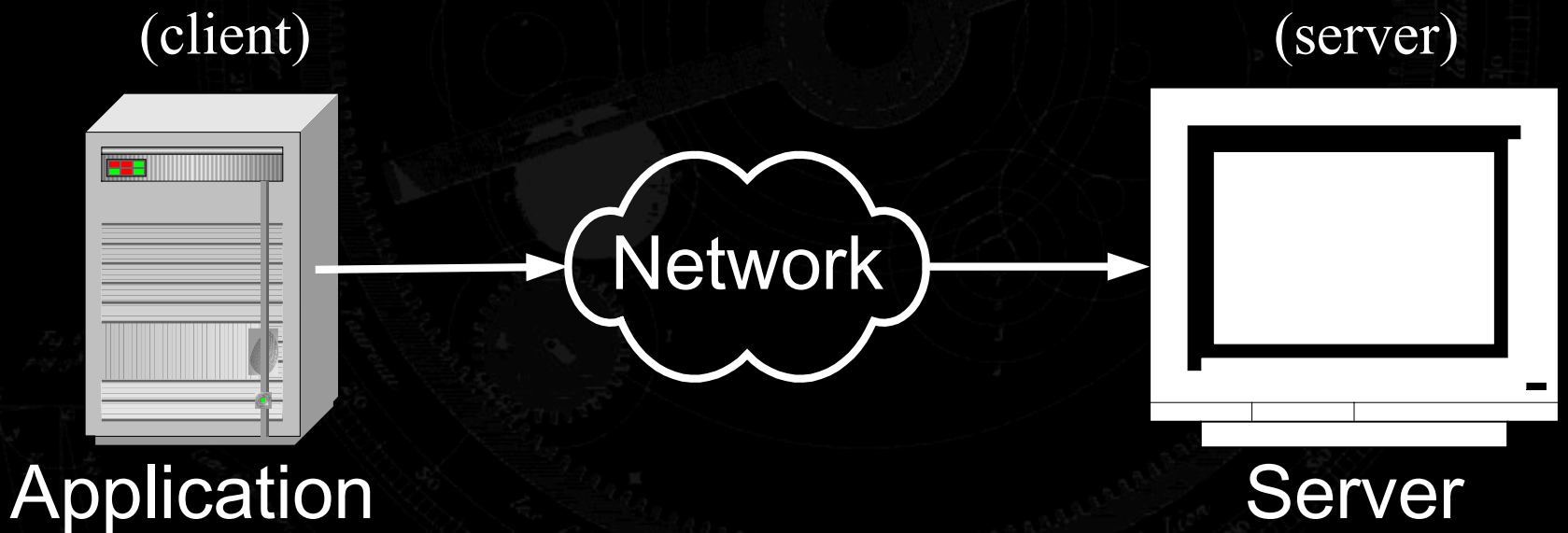
WireGL's View of Cluster Graphics



- Raw scalability is easy (just add more pipelines)
- WireGL exposes that scalability to an application

Network Graphics Streams

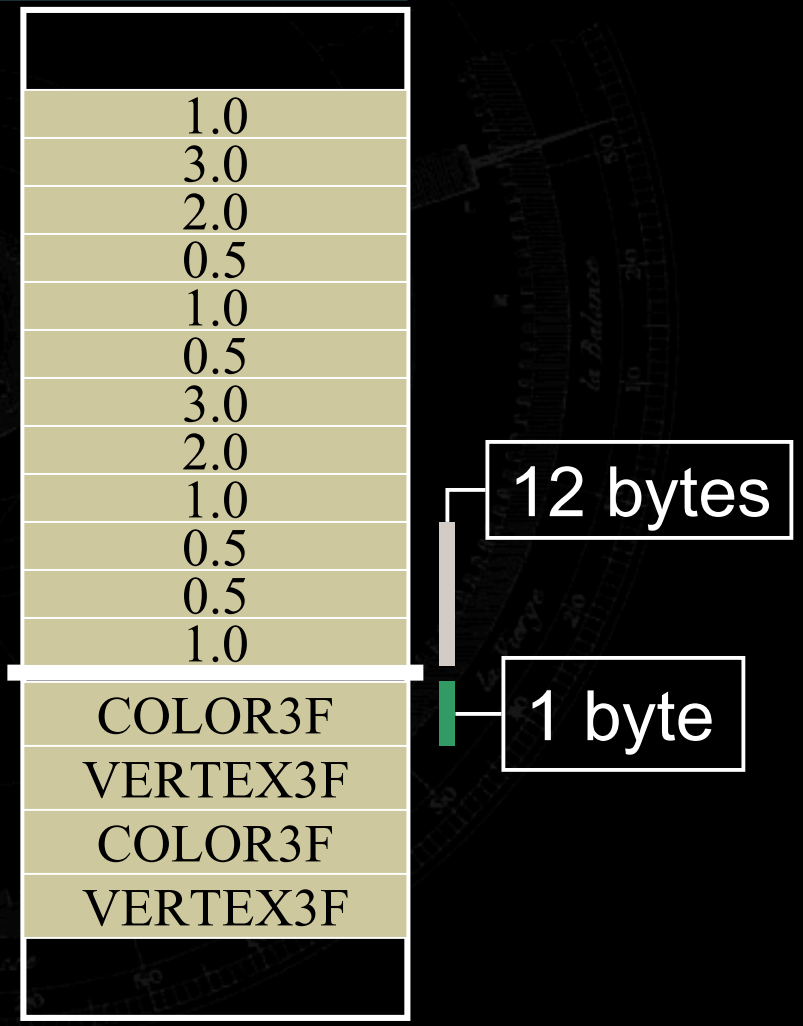
Familiar model: X, GLX, VNC, NetMeeting



WireGL Protocol

1 byte per function call

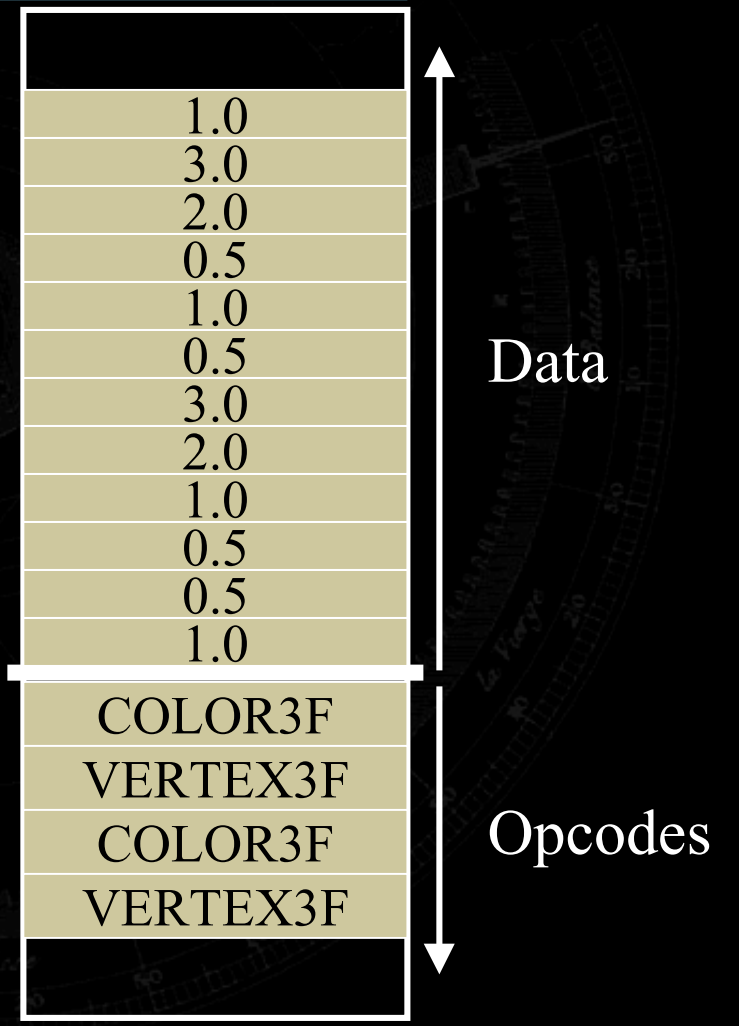
```
glColor3f(1.0, 0.5, 0.5);
glVertex3f(1.0, 2.0, 3.0);
glColor3f(0.5, 1.0, 0.5);
glVertex3f(2.0, 3.0, 1.0);
```



WireGL Protocol

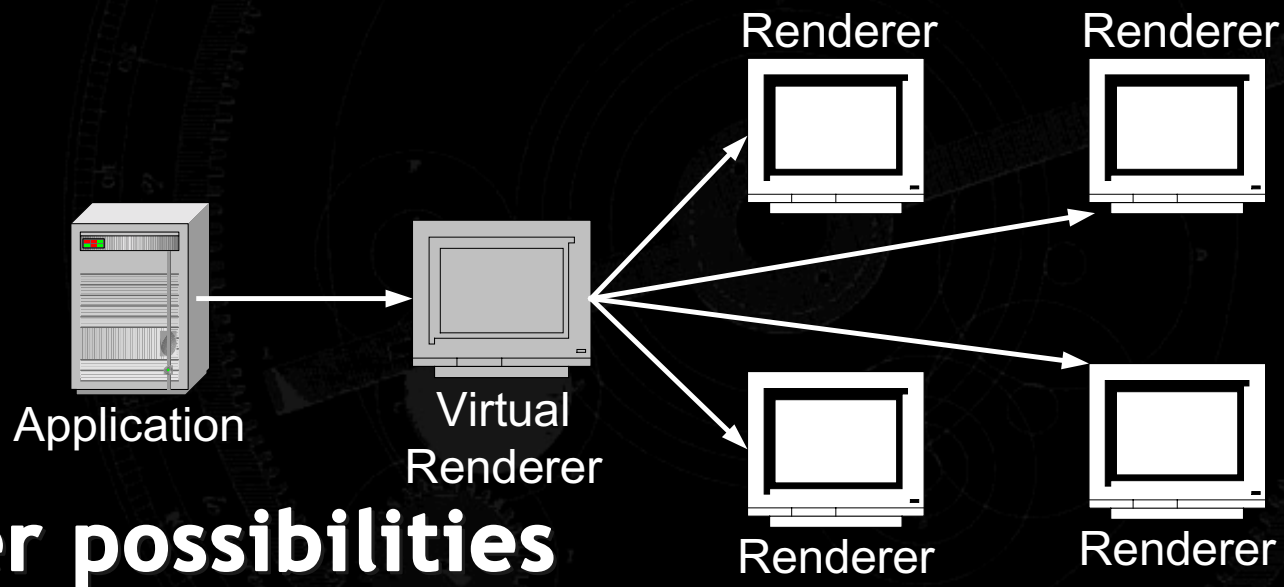
1 byte per function call

```
glColor3f(1.0, 0.5, 0.5);
glVertex3f(1.0, 2.0, 3.0);
glColor3f(0.5, 1.0, 0.5);
glVertex3f(2.0, 3.0, 1.0);
```



Driving Tiled Displays

WireGL creates a distinct stream for each server



Other possibilities

- Use network supported broadcast
- Arrange servers in a ring network

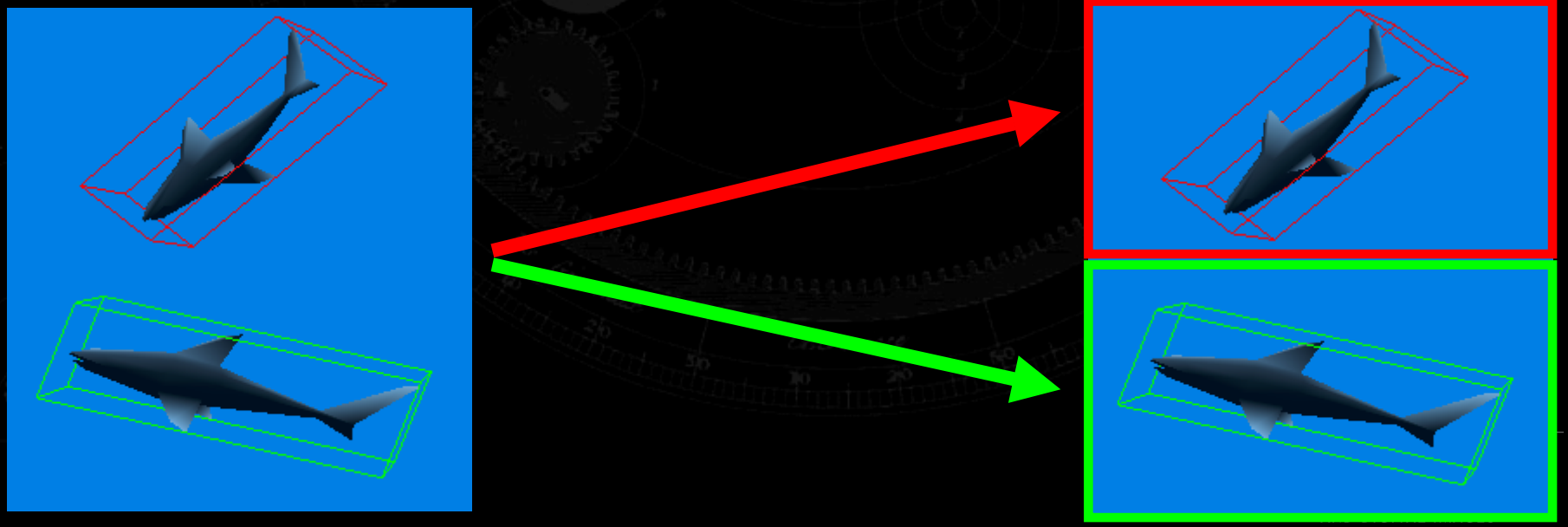
One-to-many generalizes to many-to-many

Sort-first Stream Specialization

Update bounding box per-vertex

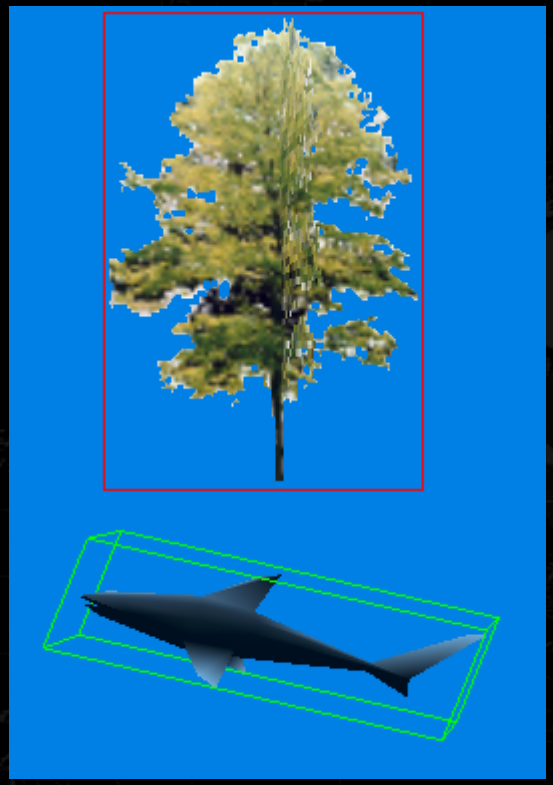
Transform bounds to screen-space

Assign primitives to tiles (with overlap)



Lazy State Updates

Only send state which is required by renderer

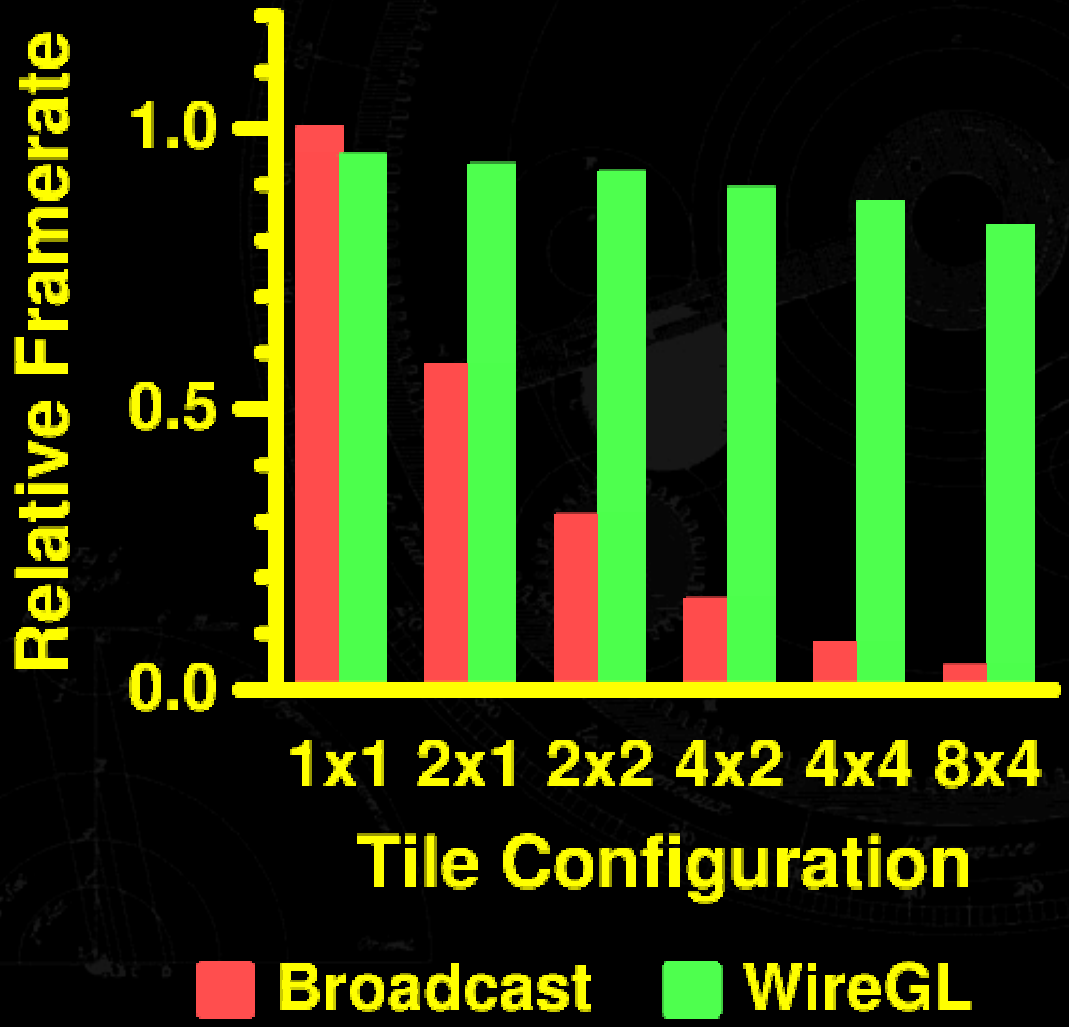


glTexImage
glBlendFunc
glEnable

glLight
glMaterial
glEnable

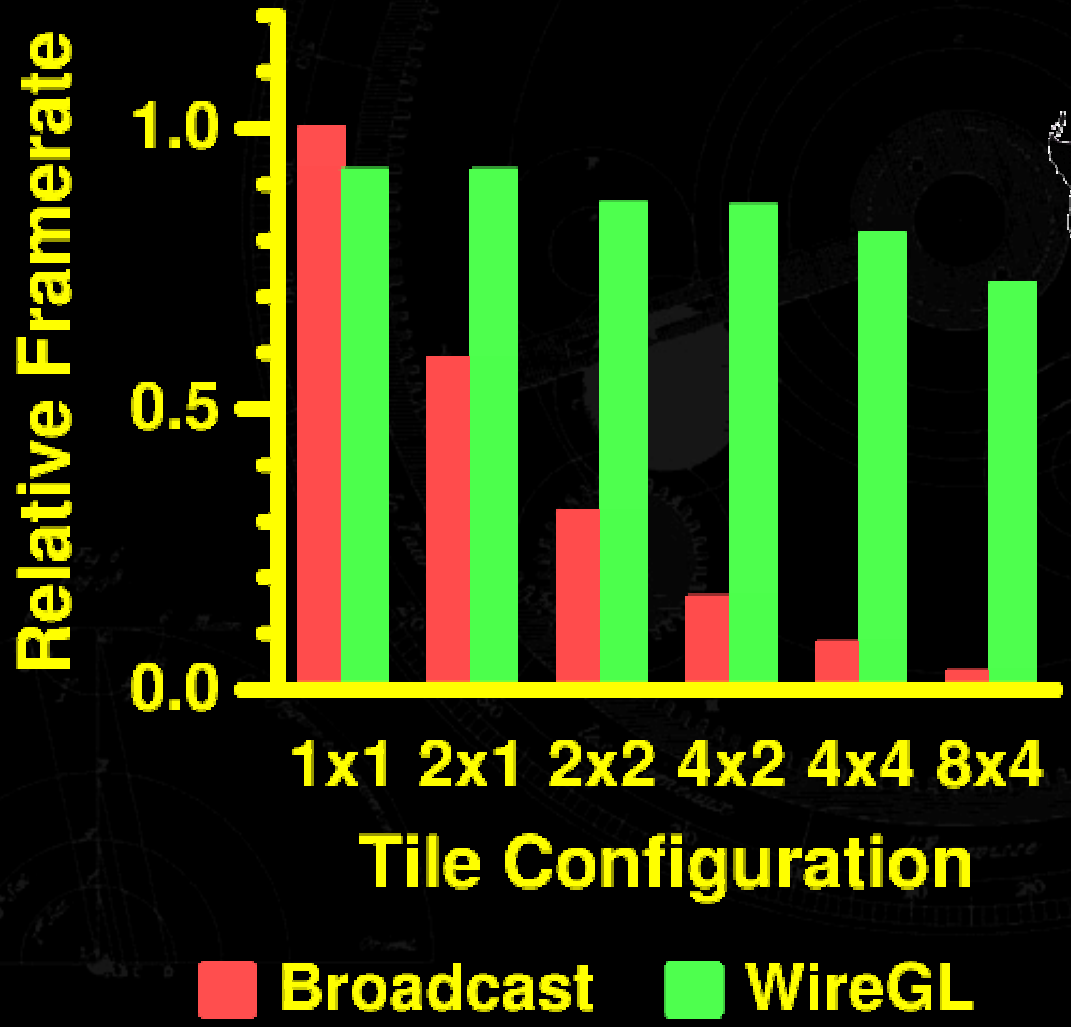
Output Scalability Results

Marching Cubes



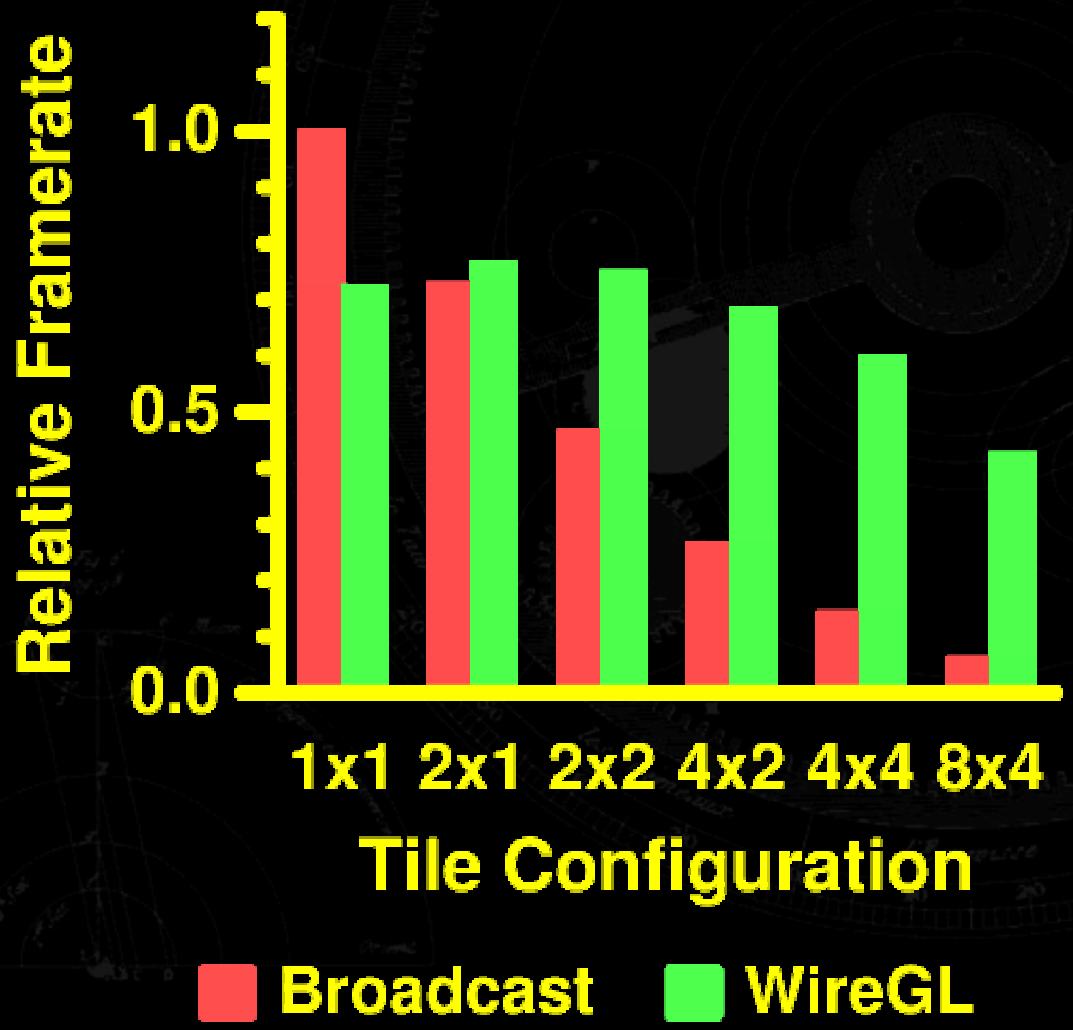
SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

Output Scalability Results



NURBS Tessellator

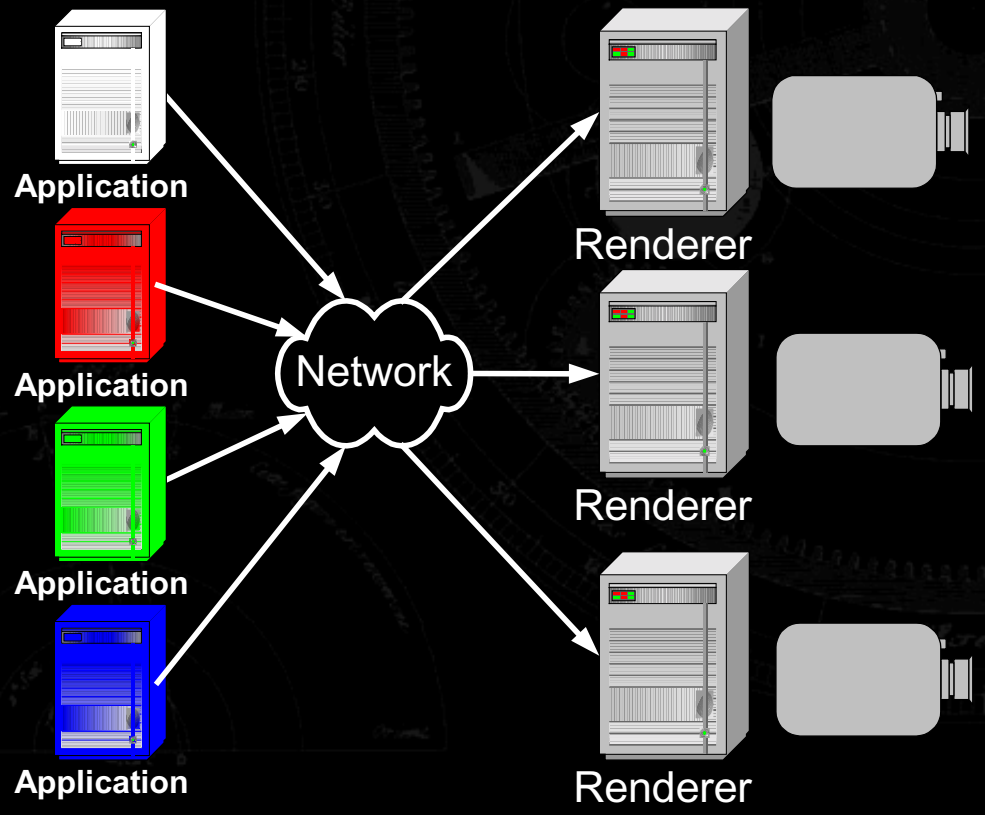
Output Scalability Results

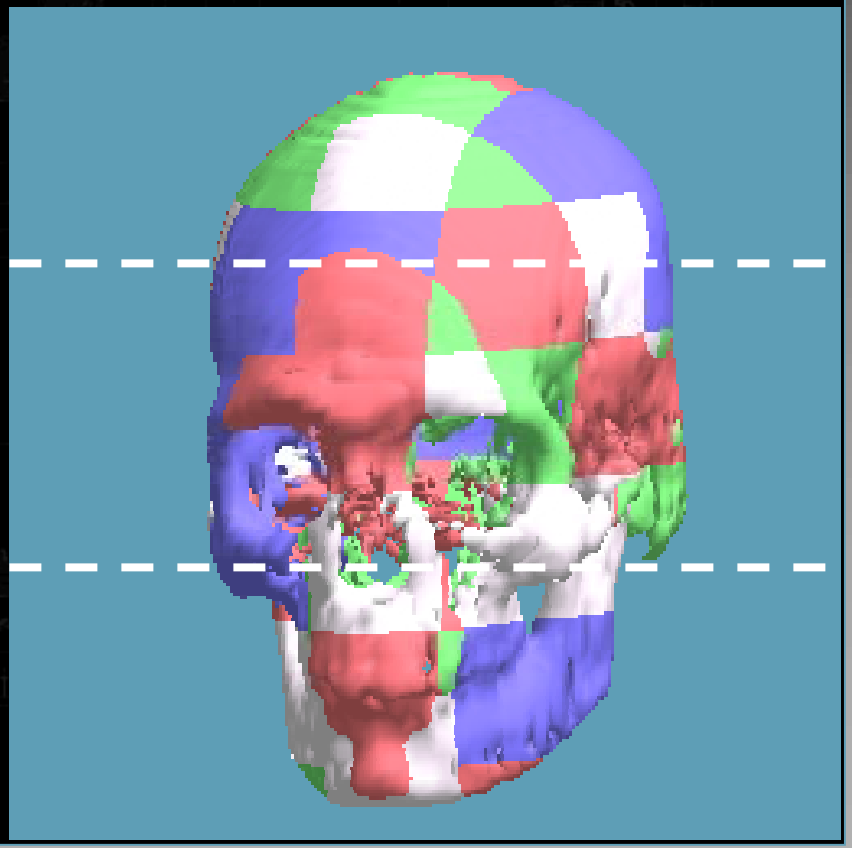


Quake III: Arena

Input Scalability

Allow multiple submitting clients
We need to order their graphics commands!





Parallel OpenGL API

Introduce new OpenGL commands:

- `glBarrierExec`
- `glSemaphoreP`
- `glSemaphoreV`

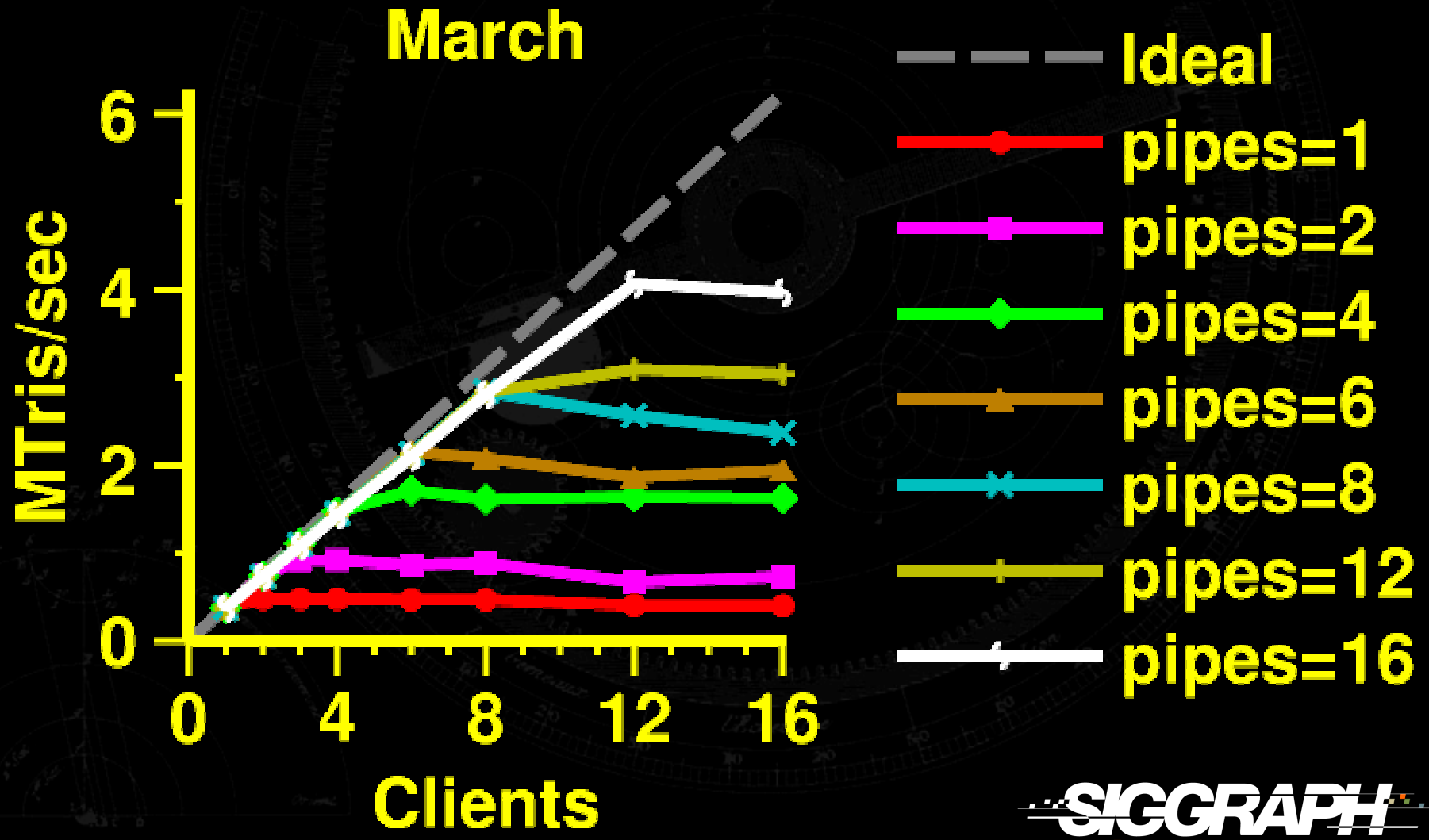
Igehy, Stoll, Hanrahan, *The Design of a Parallel Graphics Interface,* SIGGRAPH '98

Express ordering constraints between multiple independent graphics contexts

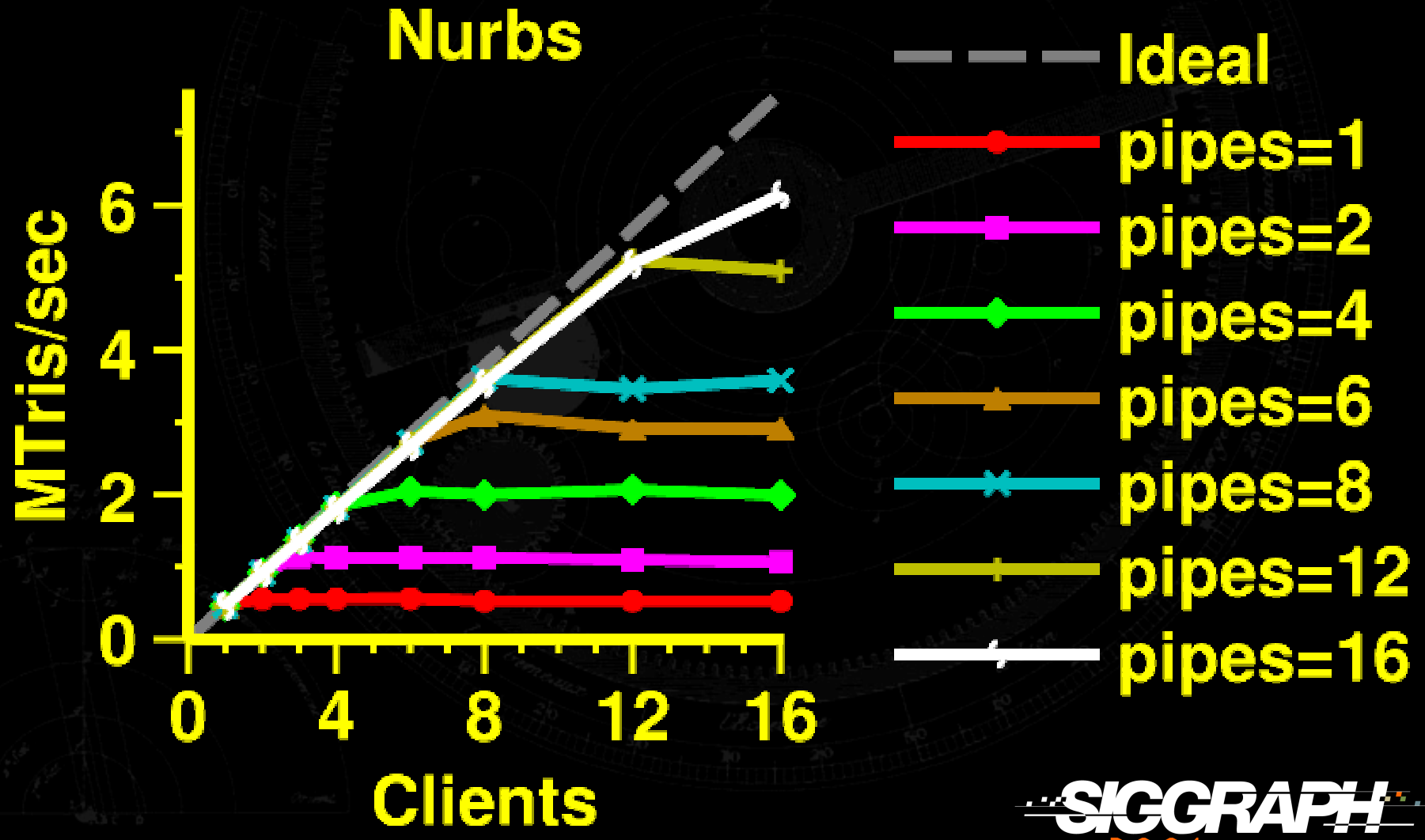
Don't block the application, just encode them like any other graphics command

Ordering is resolved by the graphics servers

Input Scalability Results



Input Scalability Results



Input Scalability Results

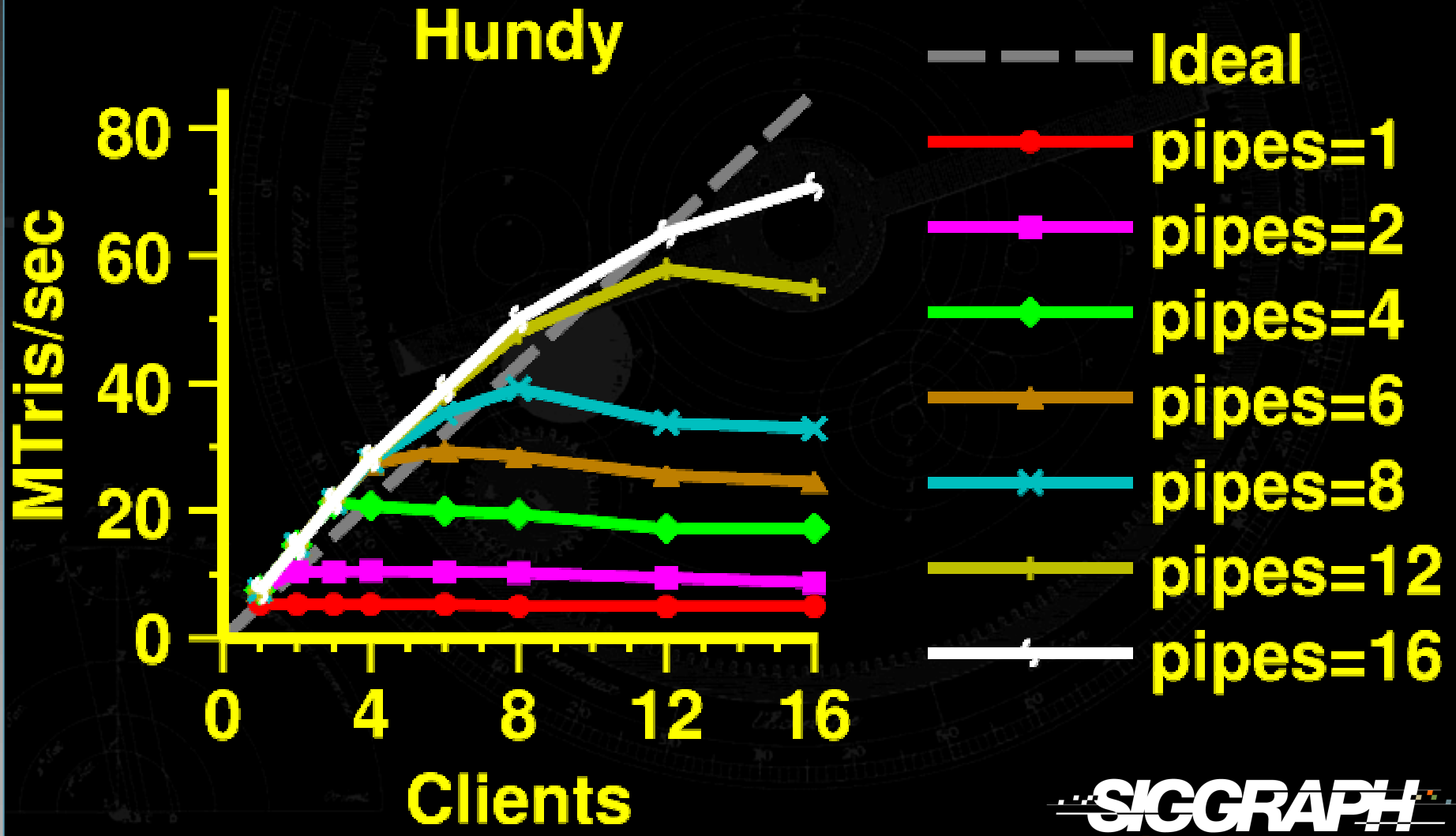


Image Reassembly

Not everyone has a large tiled display

Single tile servers limit scalability

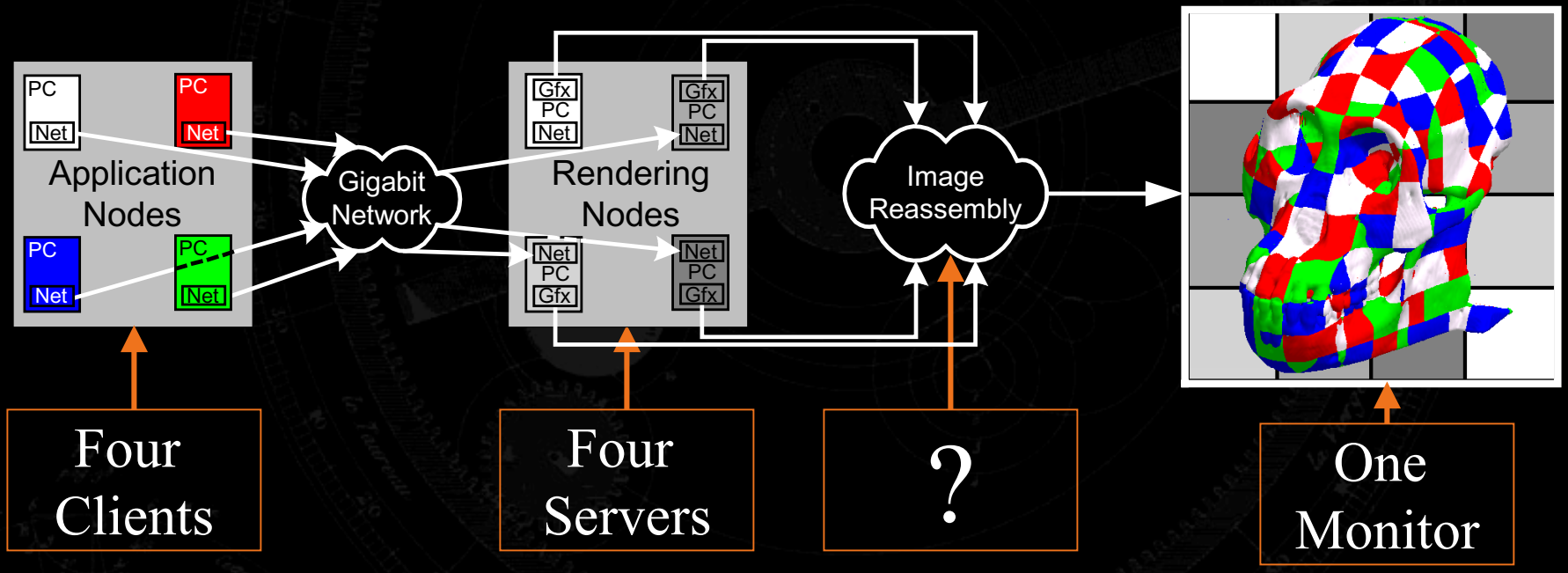
Single tile servers limit load balance

Must provide flexible image tiling

- Image tile sizes not tied to server's framebuffer
- Multiple tiles per server
- Number of tiles unrelated to actual output device

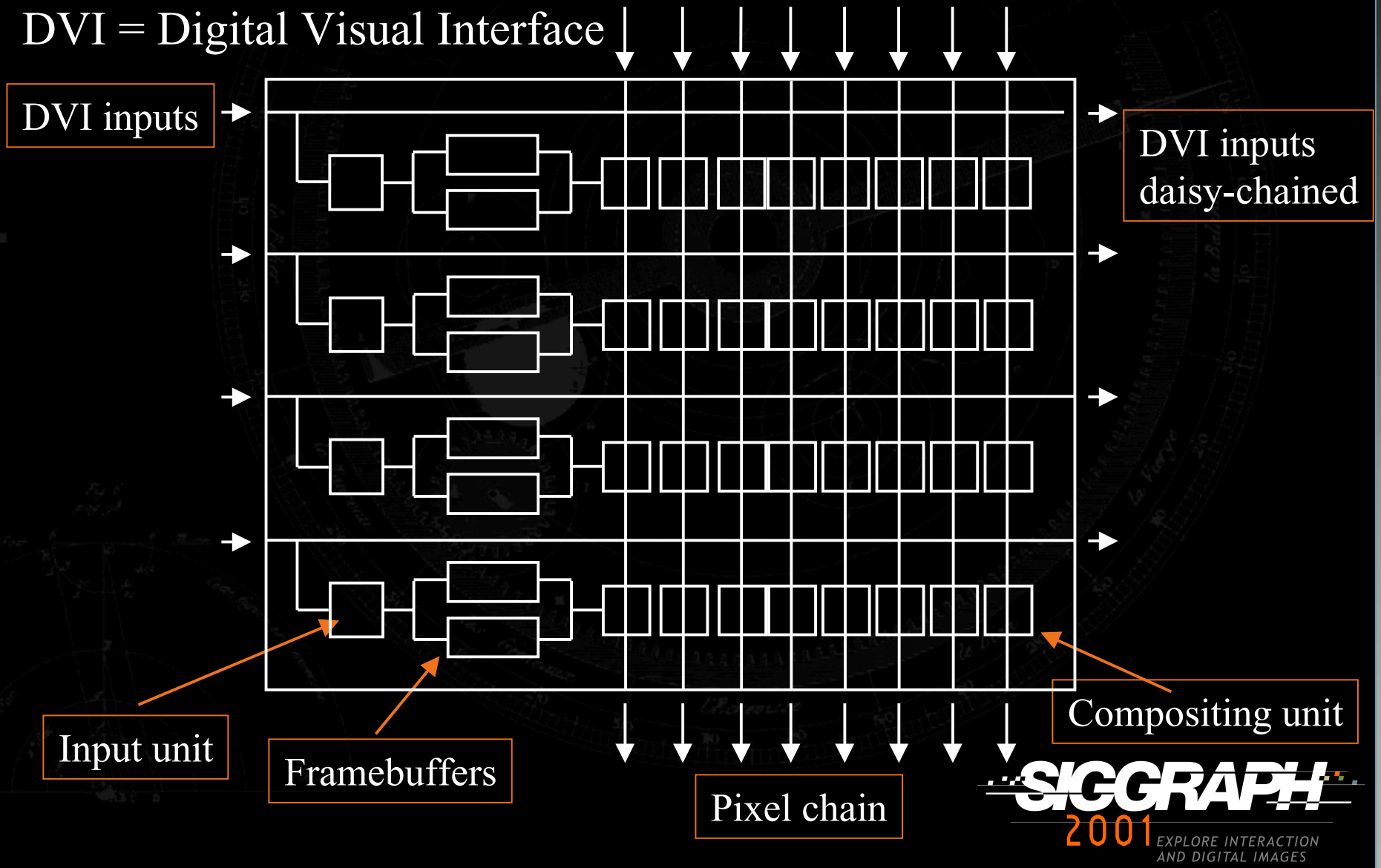
Need an image reassembly phase

A "Complete" WireGL System



Lightning-2 Module

DVI = Digital Visual Interface



Lightning-2 Prototype

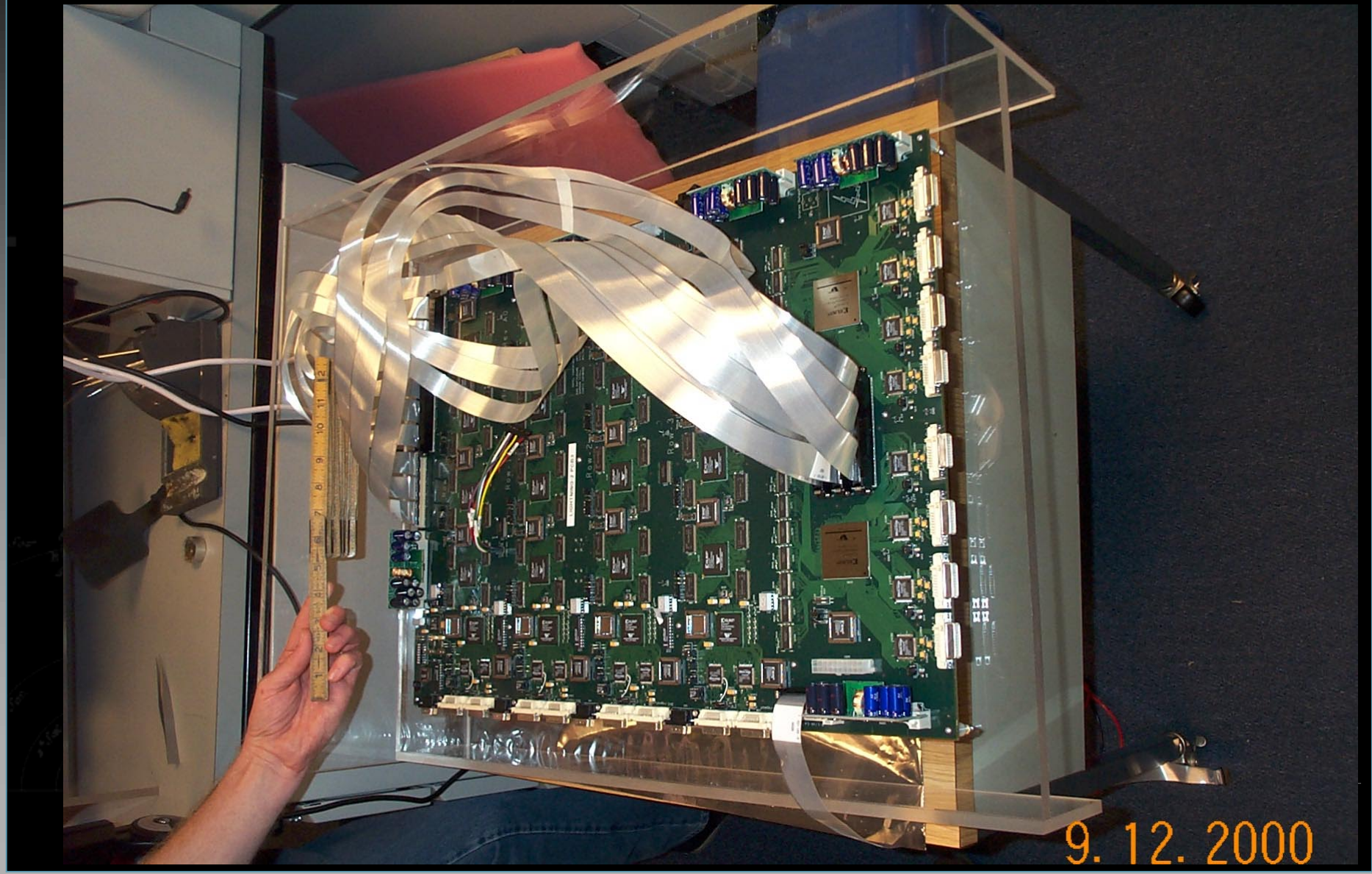
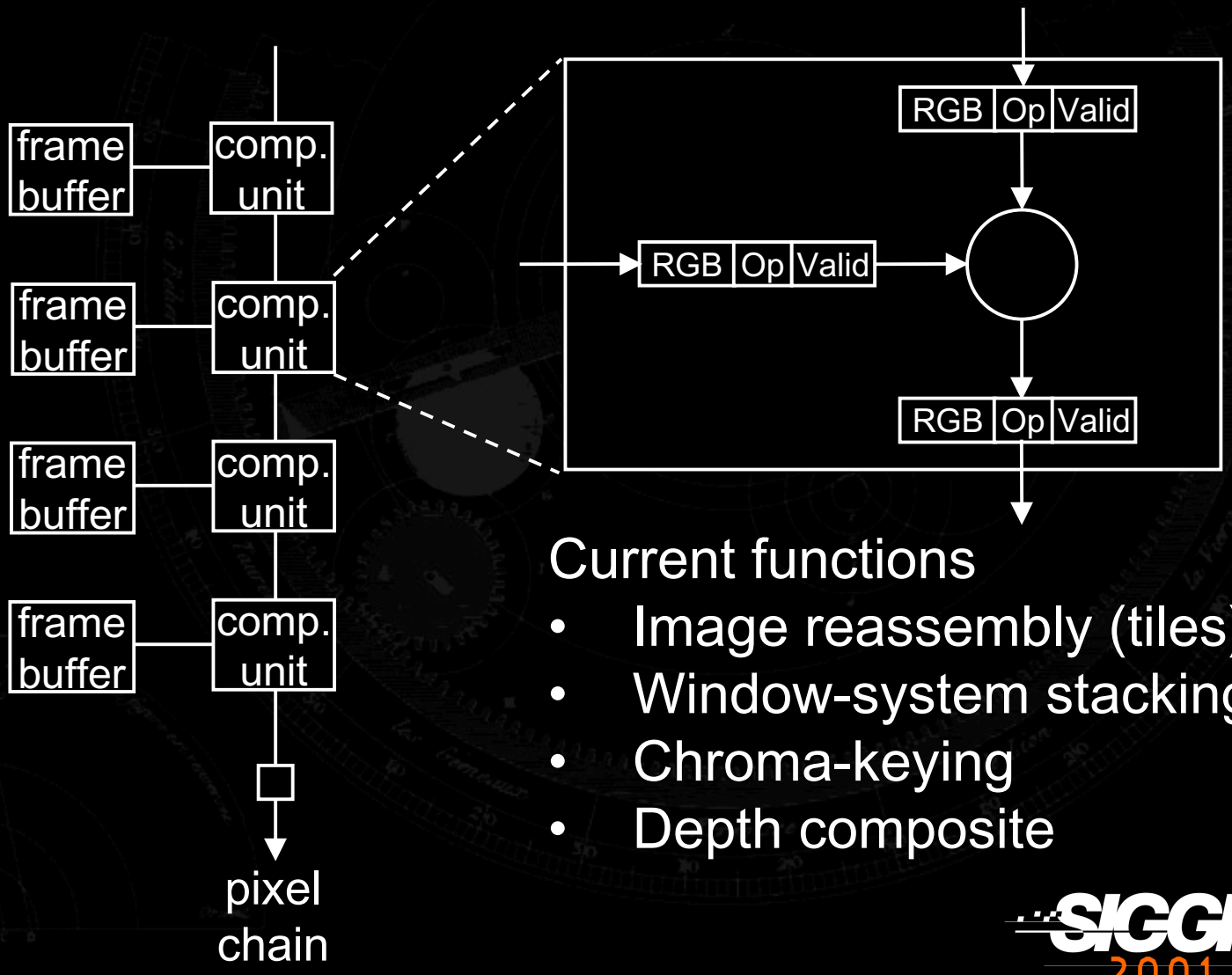


Image Composition



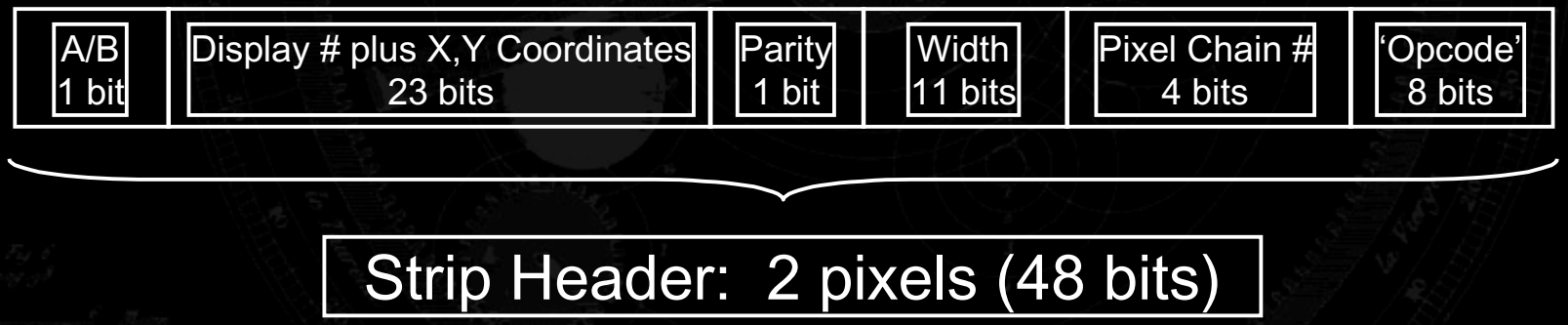
Current functions

- Image reassembly (tiles)
- Window-system stacking
- Chroma-keying
- Depth composite

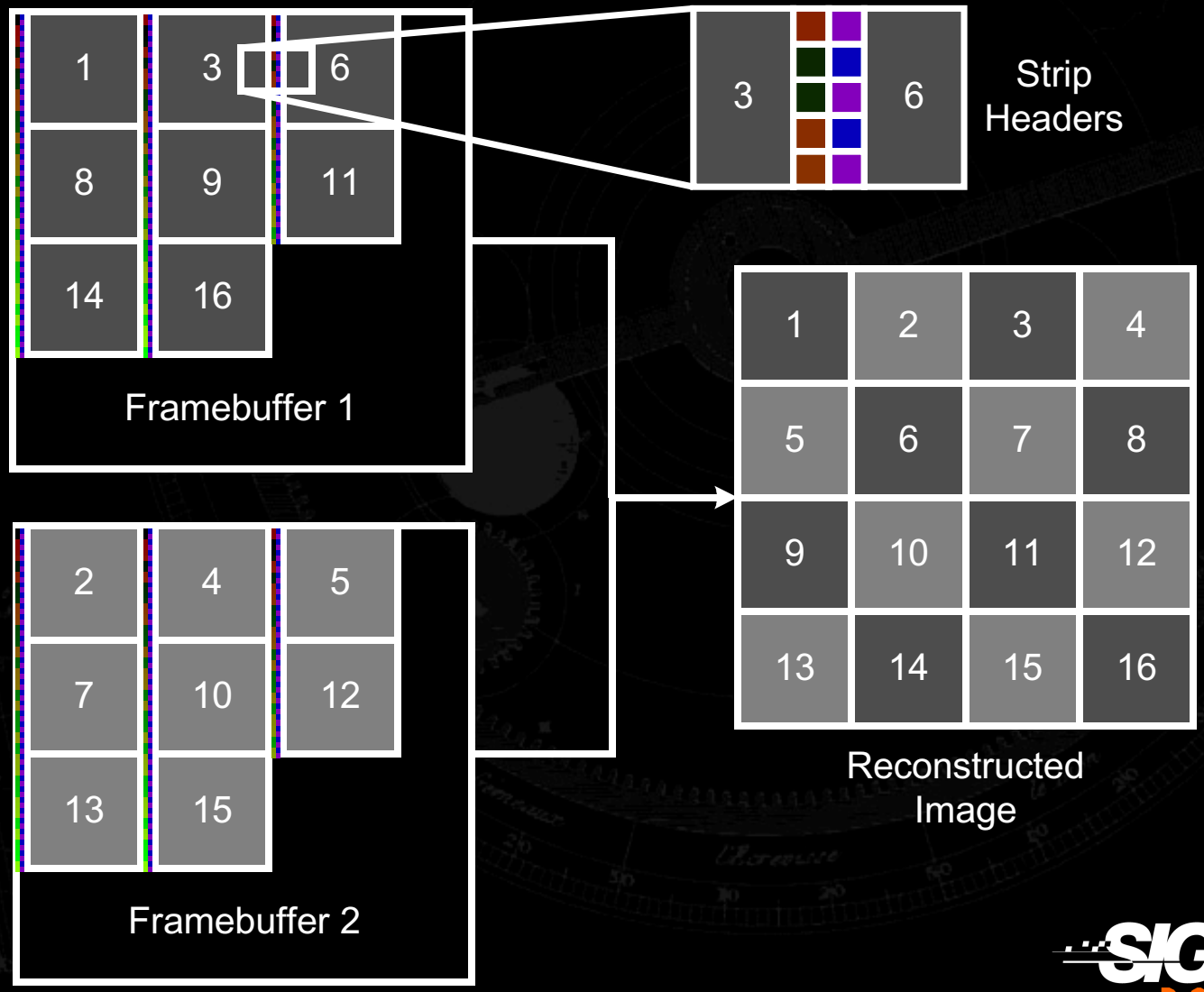
Scanline Switching

Routing information embedded in the image

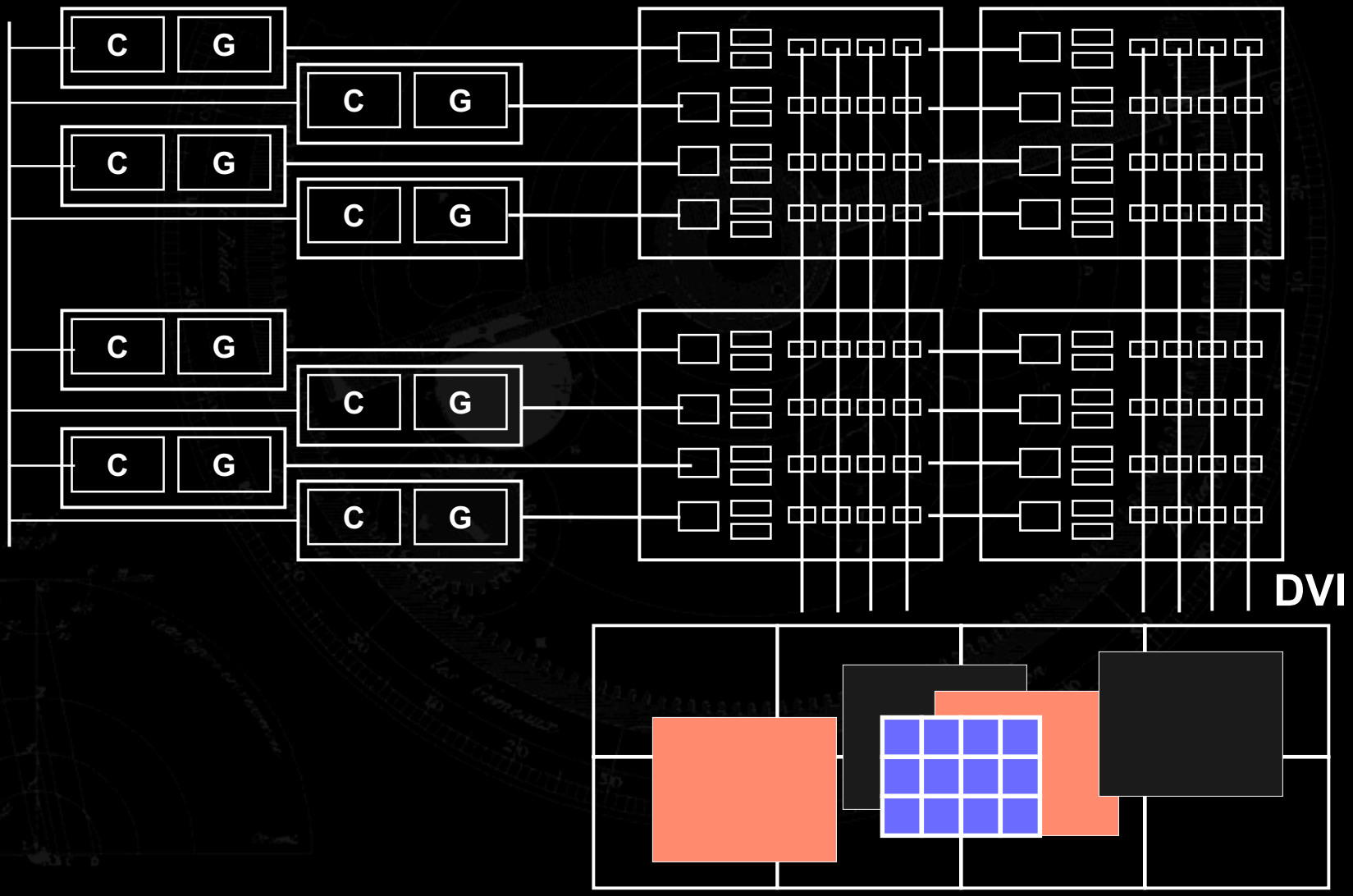
Unit of mapping is a one-pixel-high strip



Example: 16-way Tiling of One Monitor



Scalable Lightning-2



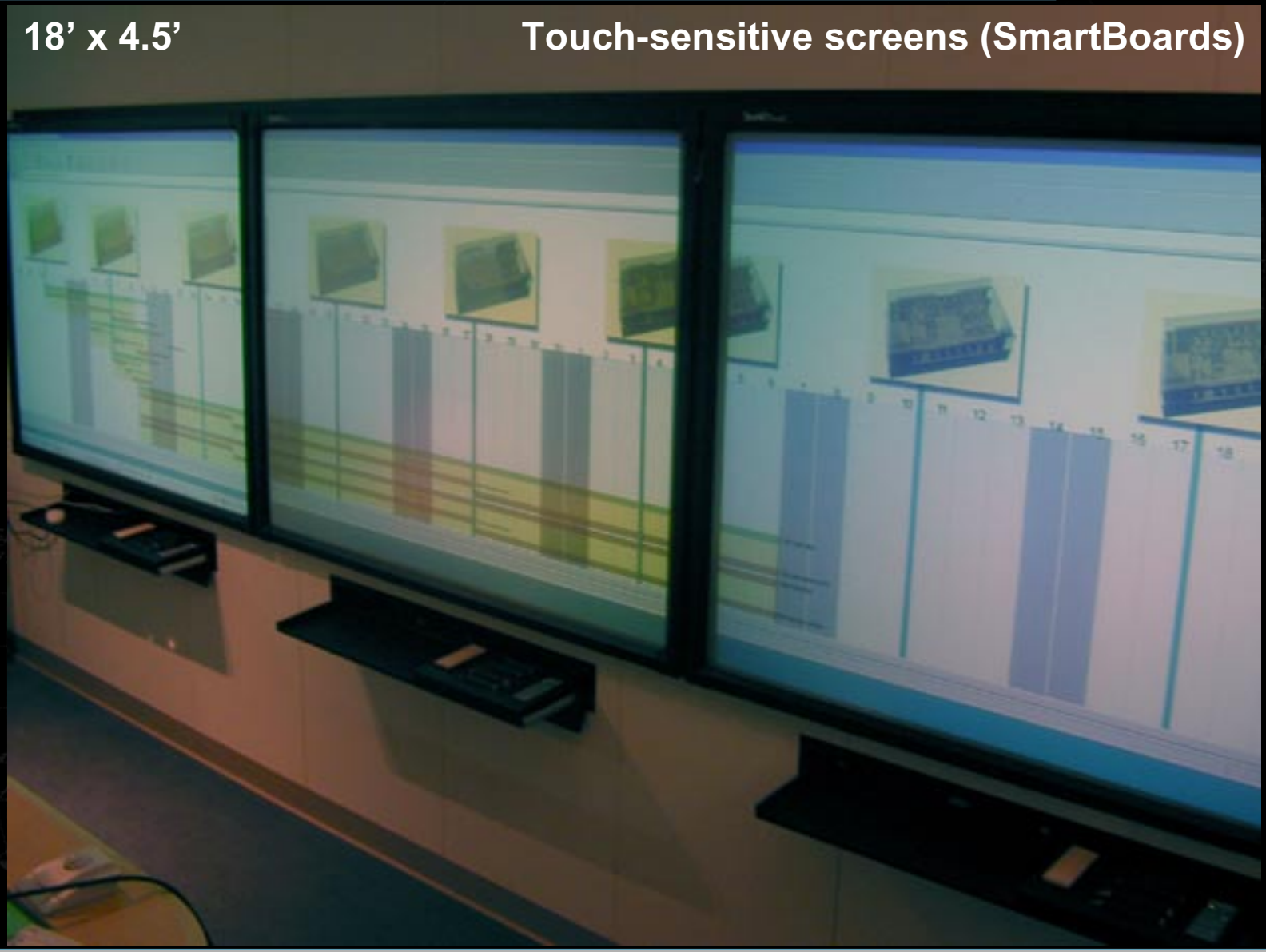
Stanford iSpace

Interactive Conference Room Table

Interactive Mural

18’ x 4.5’

Touch-sensitive screens (SmartBoards)



Interactive Mural

MultiGraphics Summary

Scalability with commodity technology

Recent breakthroughs

- Hardware:
  - Fast graphics, advanced features
  - Adequate networking
  - Core logic
- Software:
  - Transparent support for tiled displays
  - Parallel interface with ordering control

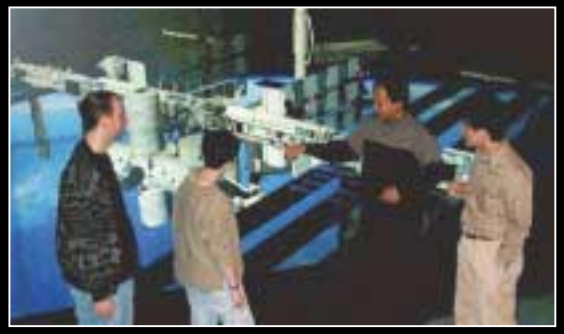
Enables large class of new applications

- Ubiquitous high performance
- Advanced display environments

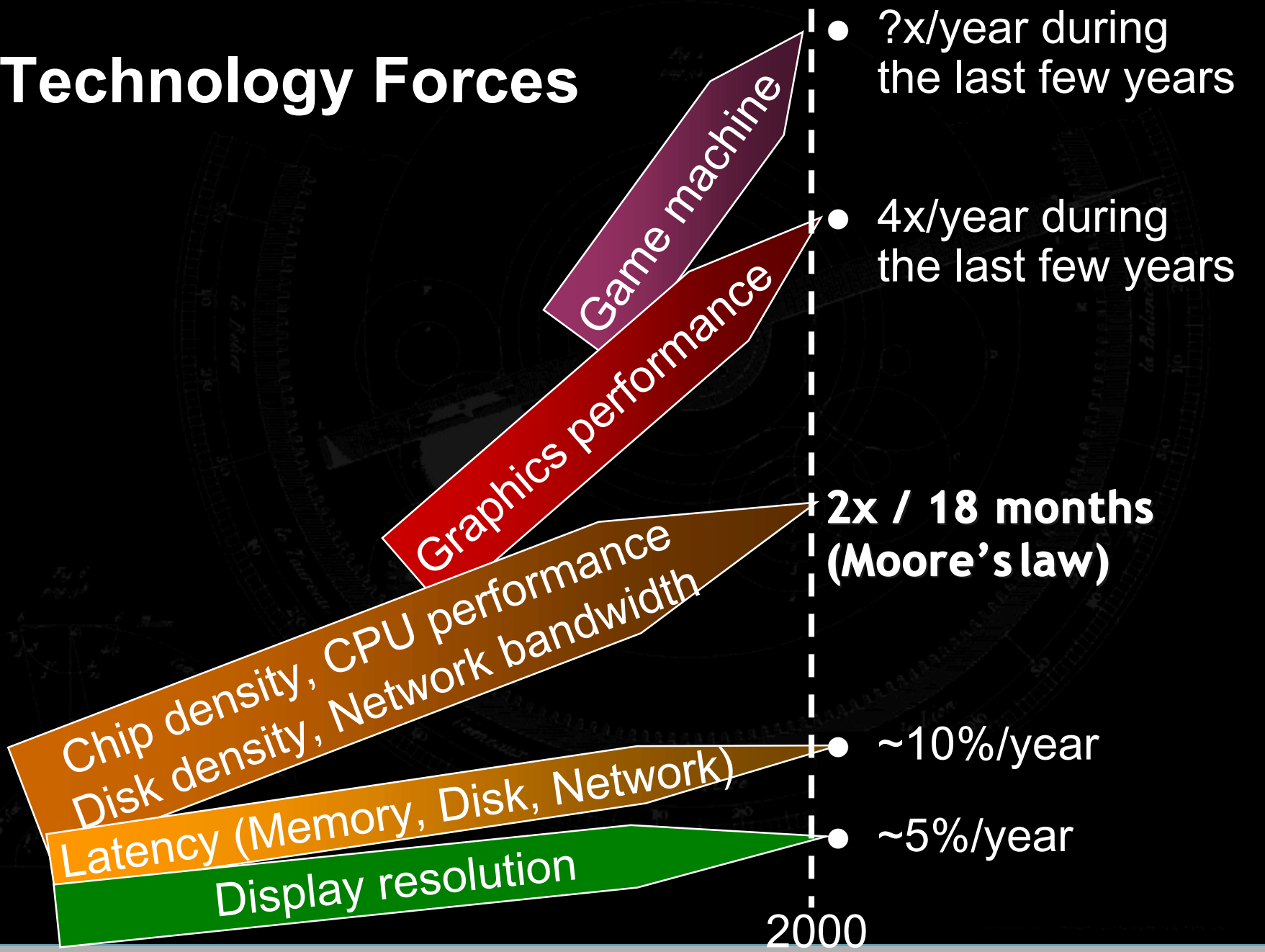
Early Experiences With An Inexpensive Scalable Display Wall System

Display Wall Project
Computer Science Department
Princeton University

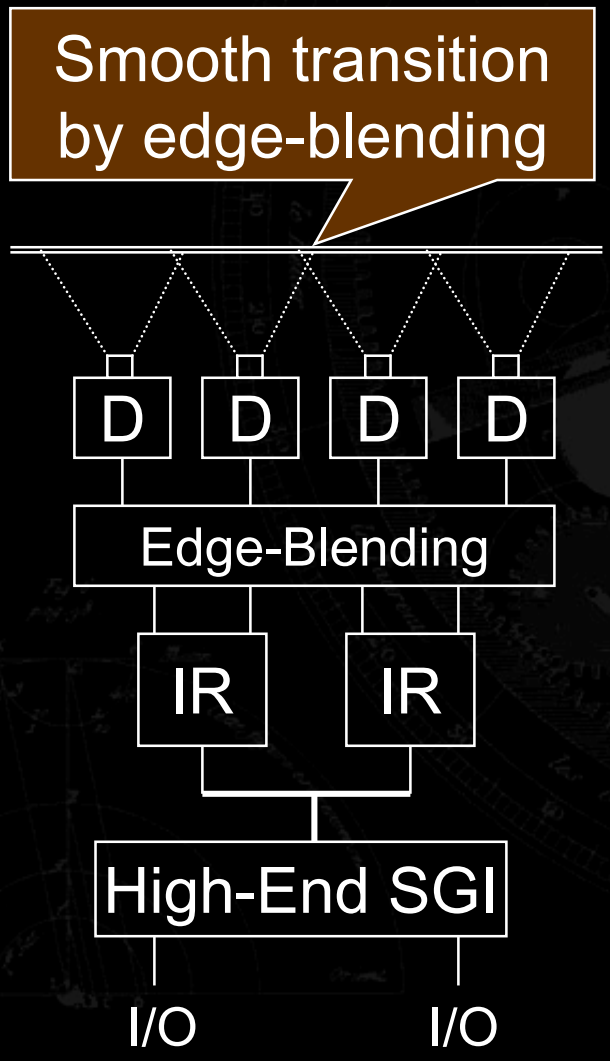
Applications



Technology Forces



Traditional Multi-Projector Display



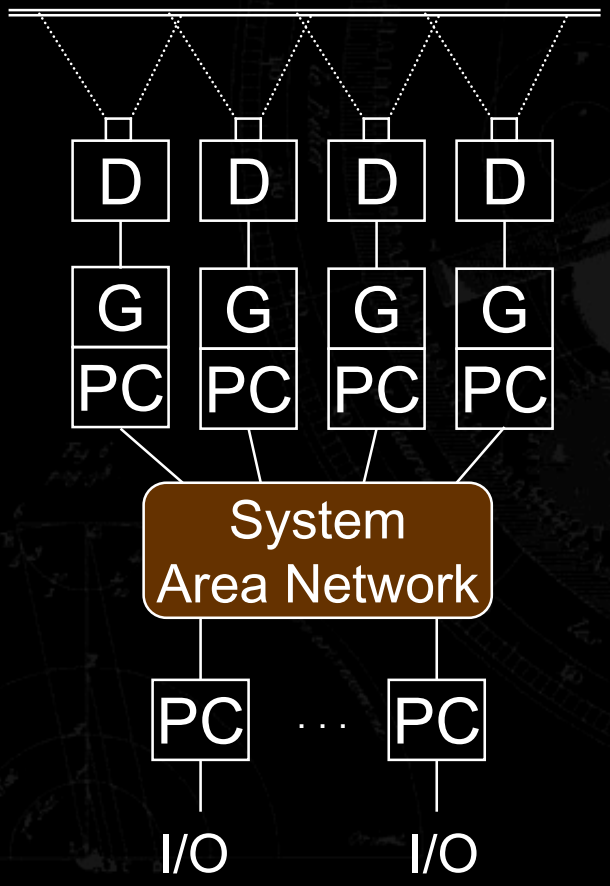
Custom or high-end components

- High-end graphics machine
- High-end graphics accelerators
- Fast internal network
- High-end projectors
- Hardware edge-blending
- Mostly front projection
- Server I/O devices

Research prototypes

- PowerWall (U. Minnesota)
- Same as above, but rear projection without edge-blending

Princeton Display Wall



Multiplicity of commodity parts

- Commodity PCs
- Cheap 3-D graphics accelerators
- System area network
- Portable LCD/DLP projectors
- Software/optical edge-blending
- Rear projection
- Commodity PC I/O devices

Advantages

- Inexpensive
- Track technology well

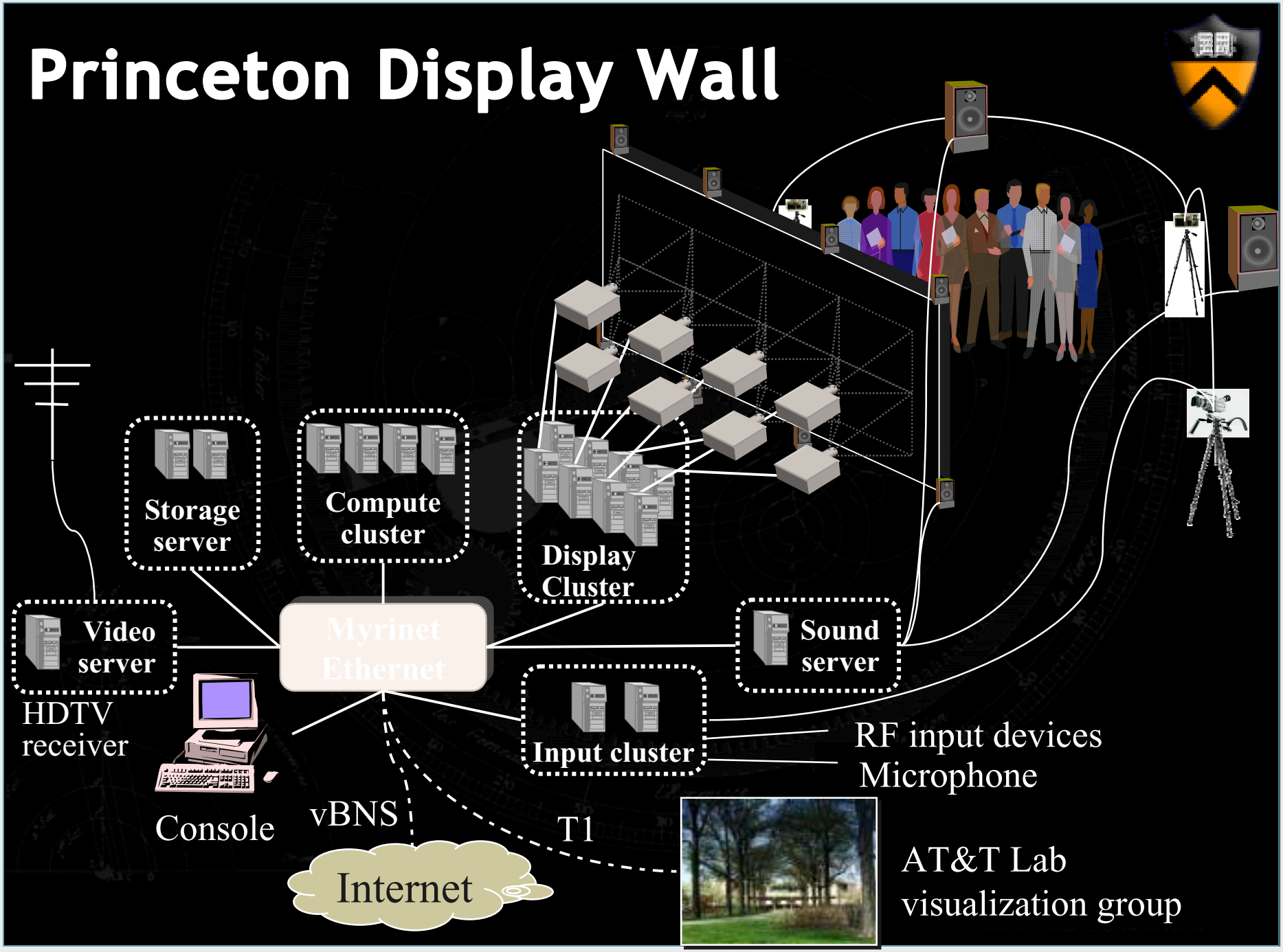
Research Challenges

Construction

- Scalable resolution display
- Scalable, inexpensive architecture
- Software tools for existing and new applications
- Applications

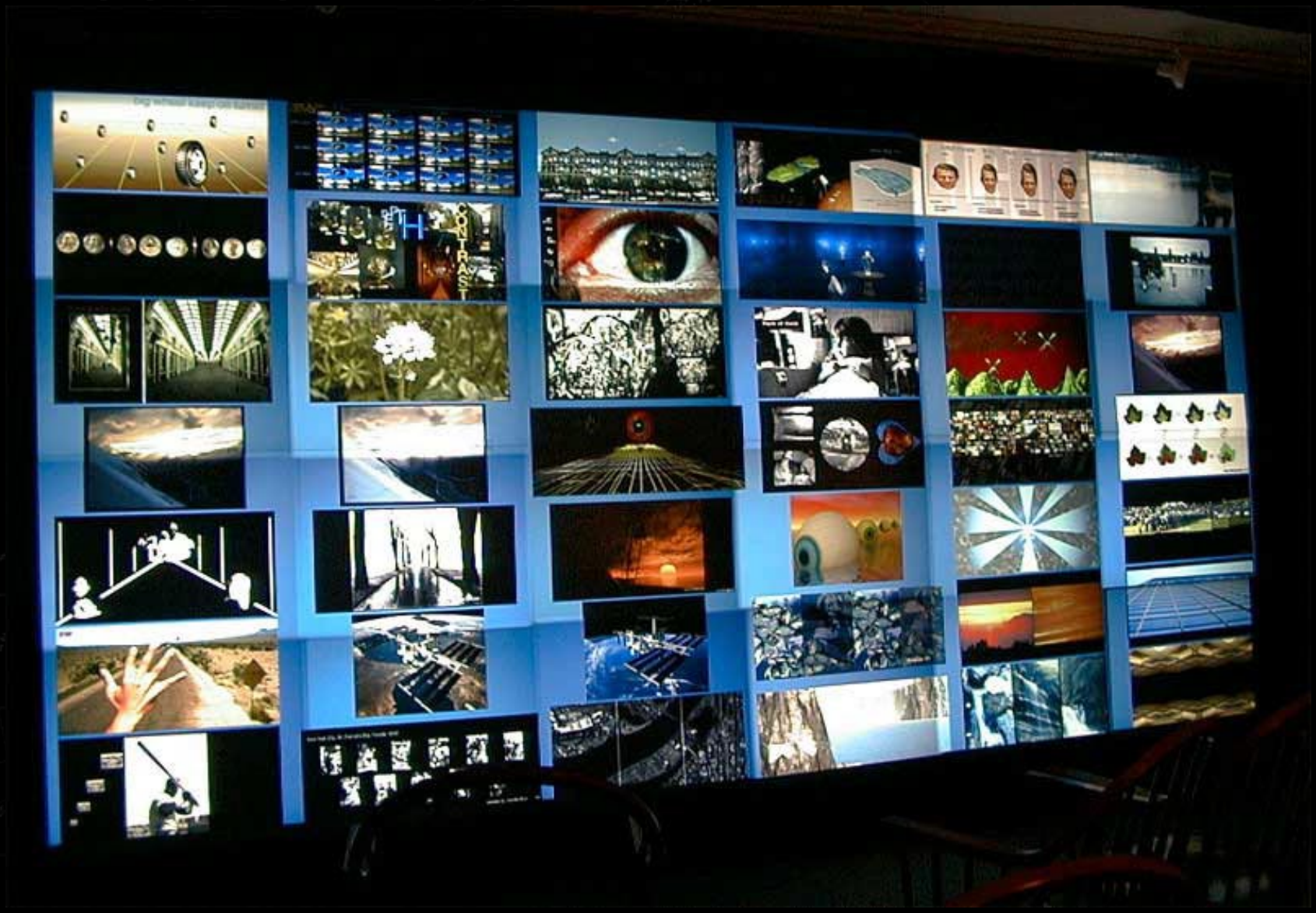
Usability

- Natural user interfaces for scalable display systems
- Guidelines for visualization on large-format displays
- Scalable resolution content


Princeton Display Wall
Storage server
Compute cluster
Display Cluster
Video server
Myrinet Ethernet
Sound server
HDTV receiver
Input cluster
RF input devices
Microphone
Console
vBNS
T1
Internet
AT&T Lab visualization group

Behind the Wall

A Recent Photo

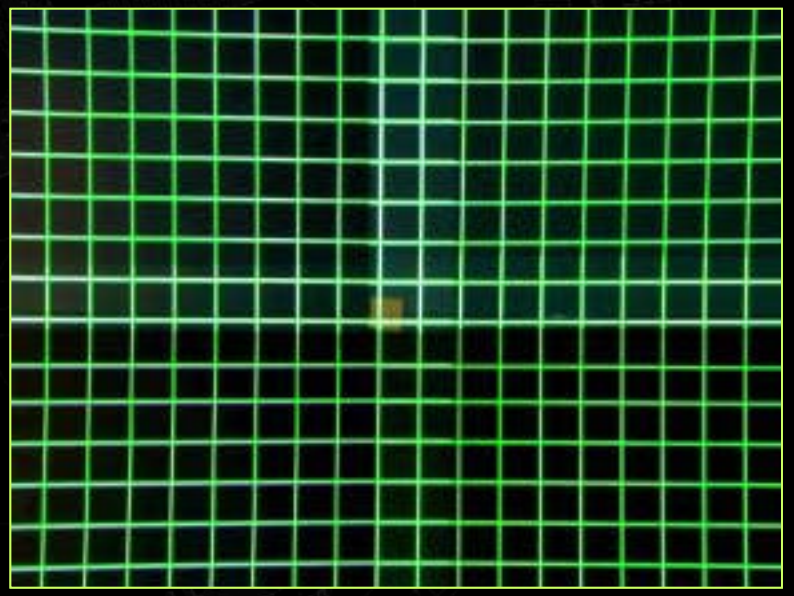


Automatic Alignment
(Y. Chen, A. Finkelstein, D. Clark, K. Li)

UNC and MIT approaches require calibrated cameras to do image transformation

Our approach

- Use one or more cameras to detect misalignments
- Adjust projective matrix iteratively
- Pre-warp projected imagery



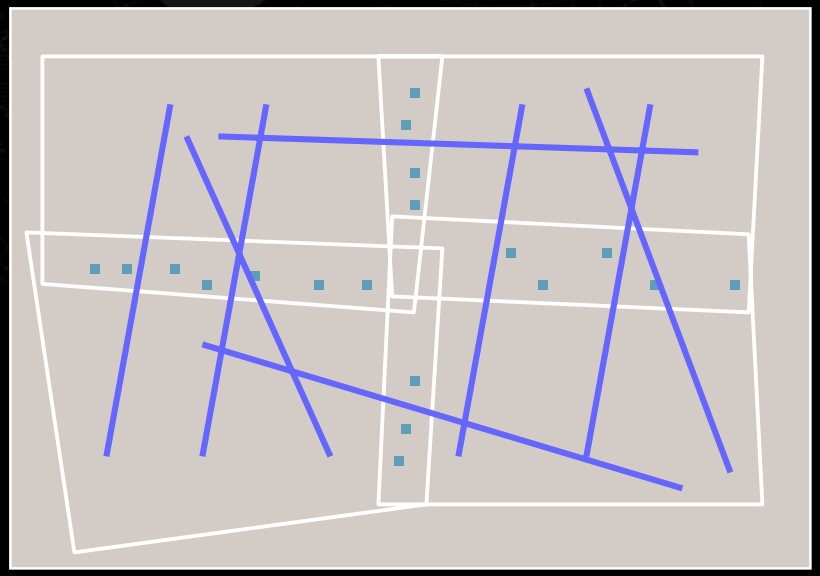
Automatic alignment

Uses inter-projector geometric constraints

very much like rigid-body constraints

point matches ⇒ bolts

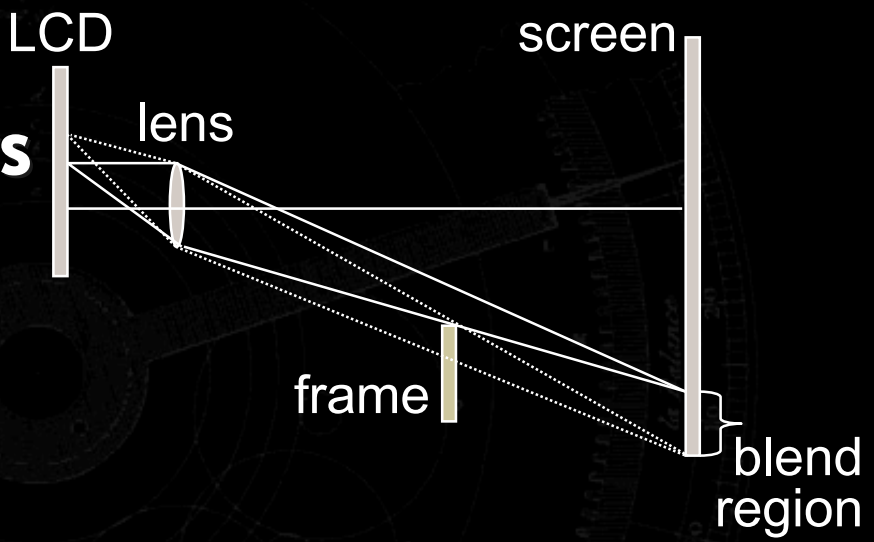
line matches ⇒ struts



Edge Blending

Problems with projectors

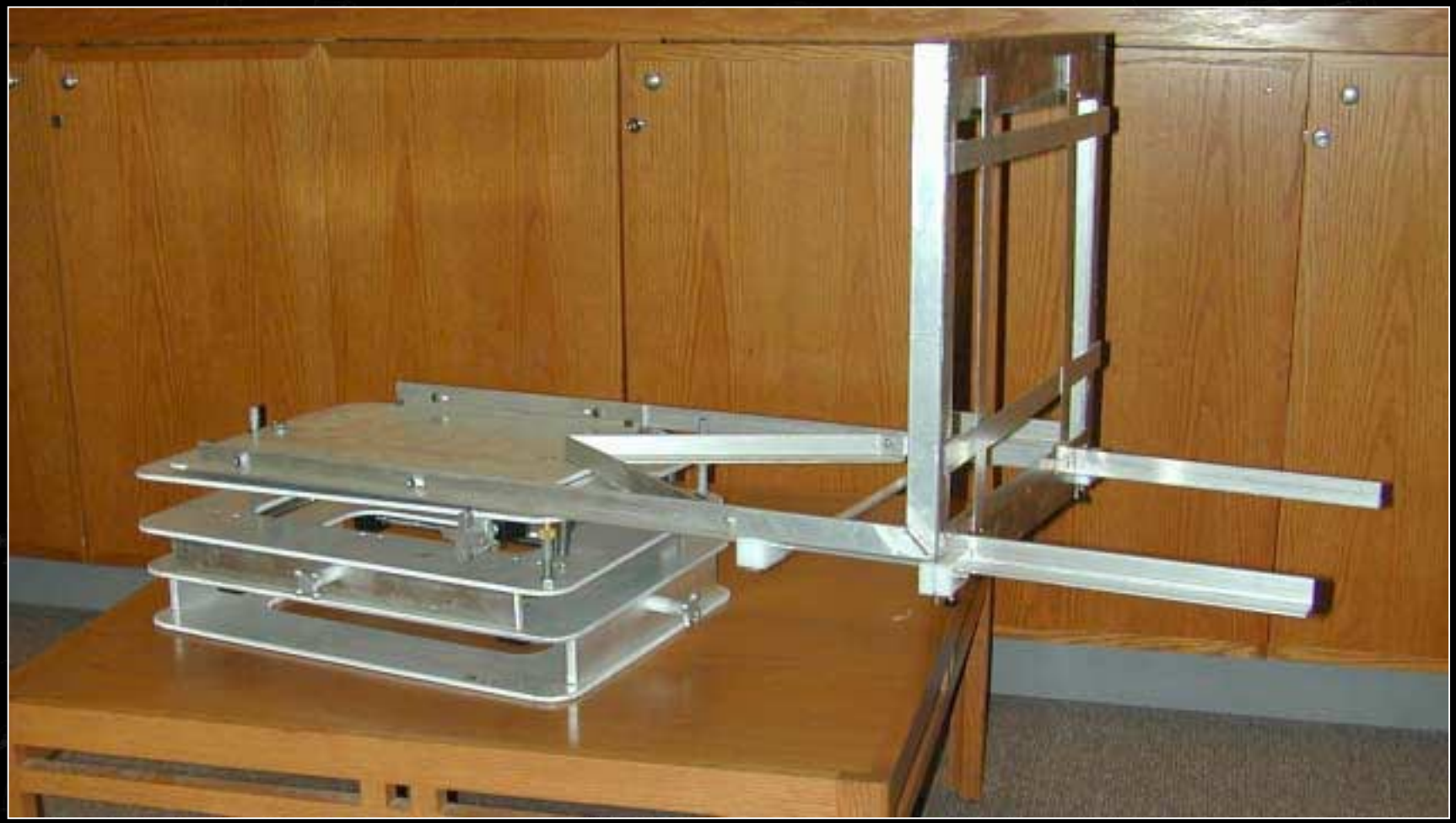
- black + black = gray
- bad pixels at the boundary of the projected image



Solution

- Use *aperture modulation* to do optical blending
- A blending frame on the projector mount
- Option: use uncalibrated camera feedback to do tuning for better blending

Blending: projector mount



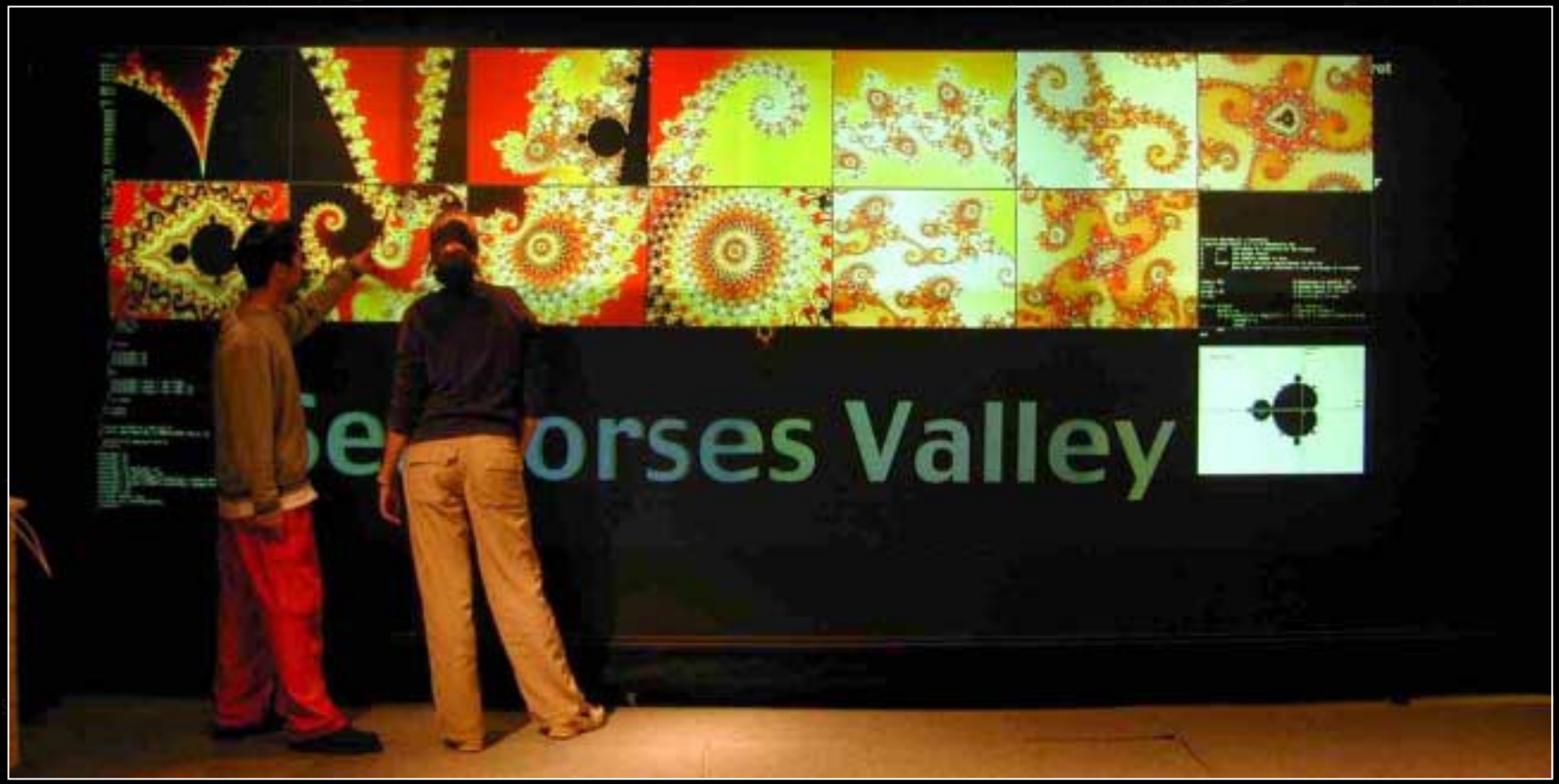
Tools for Porting Applications
(Y. Chen, Z. Liu, T. Funkhouser, R. Samanta)

Custom-designed (Still Image Viewer)
- Distributed server and command controller

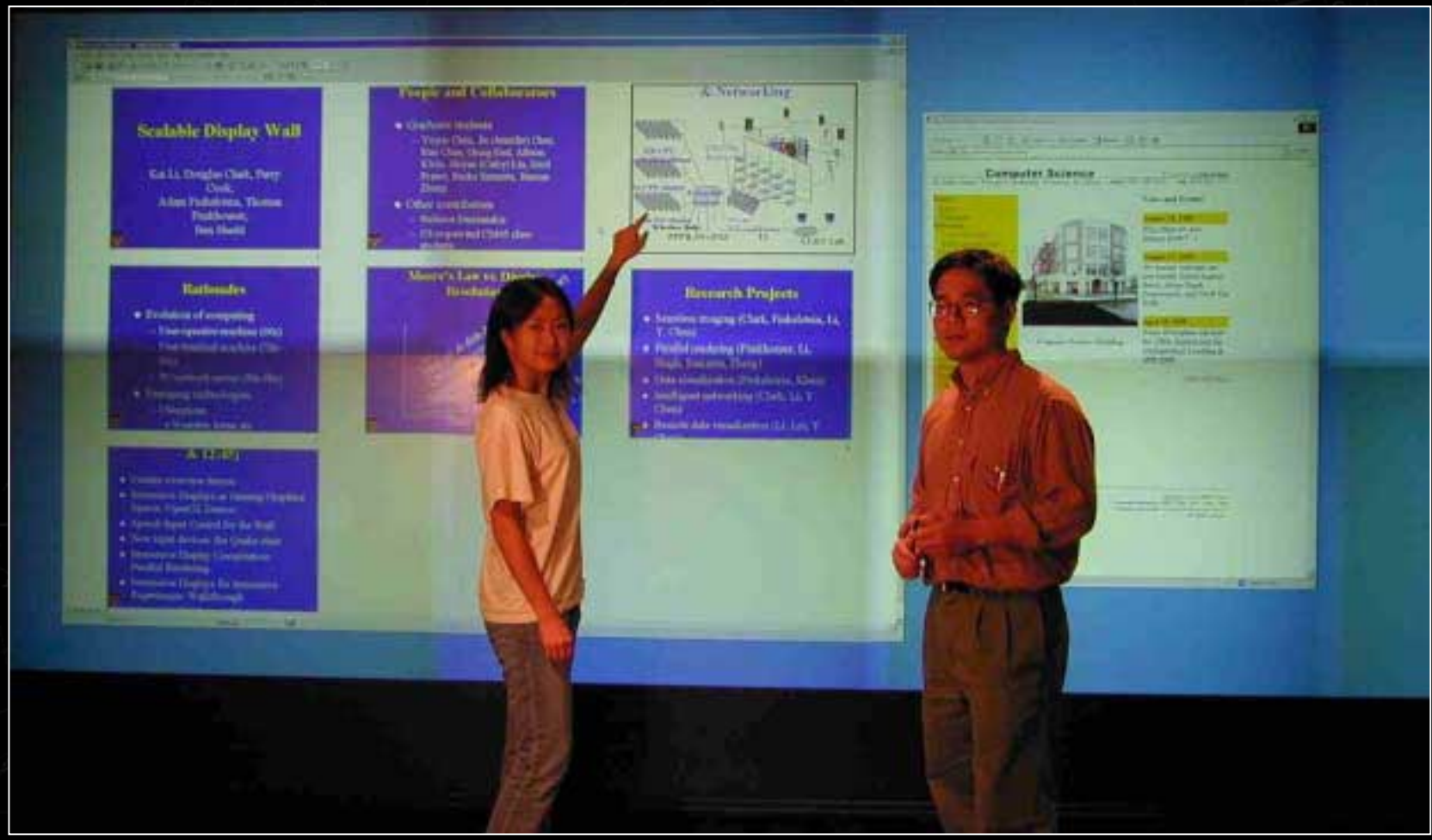
Distributed primitive approach
- OpenGL: Intercept at DLL level and execute remotely
- VDD: Intercept at device driver and execute remotely

Synchronized programming model
- Runtime synchronization (Building Walkthrough)
- System call synchronization
- Replicate and update camera and perspective

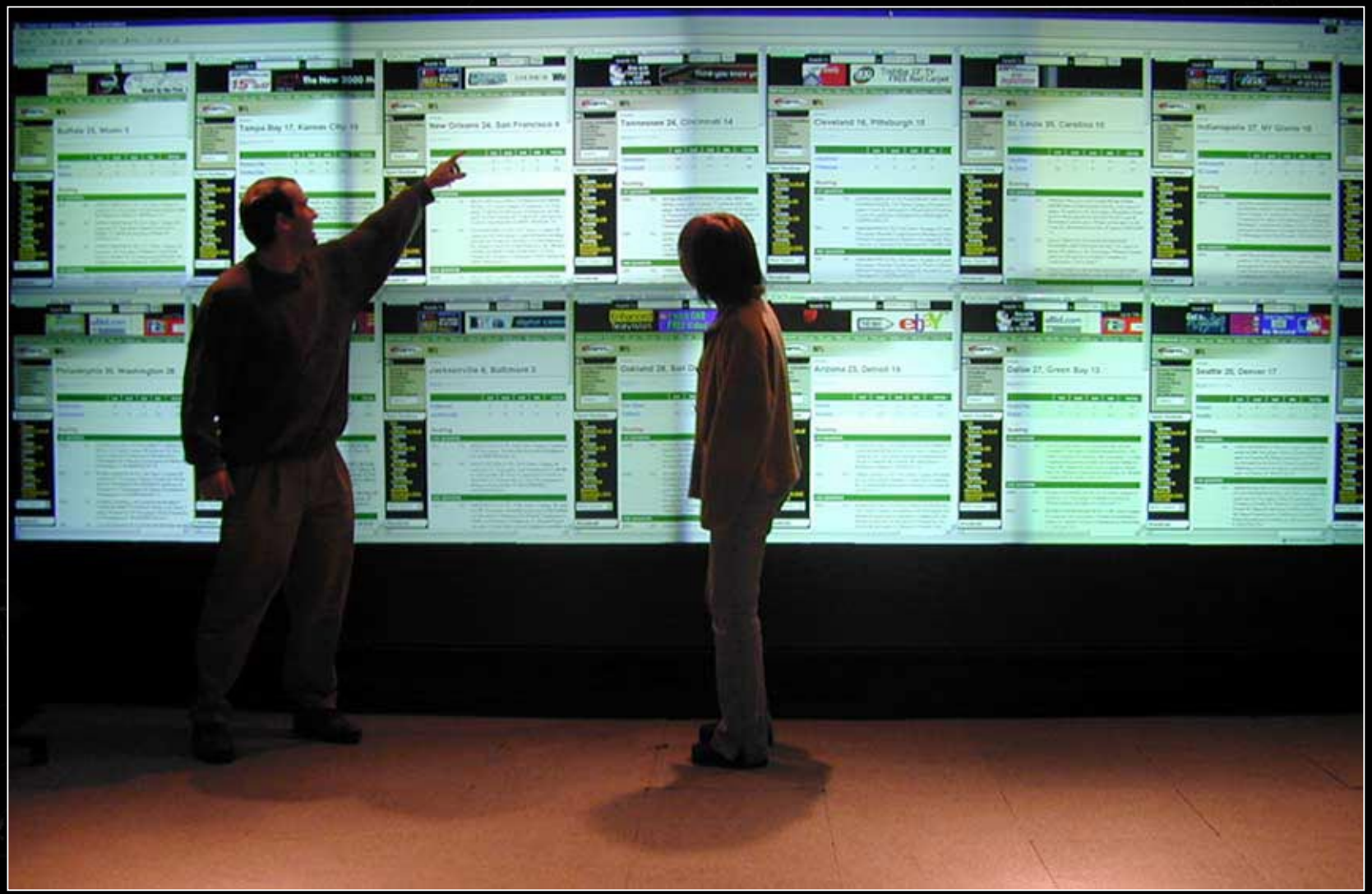
Exploring the Mandelbrot Set
(W. Kidd)



Virtual Display Driver
(Y. Chen, Z. Liu)



Multibrowser(A. Filner)



Synchronized execution example

Walkthrough of Soda Hall

Parallel Rendering

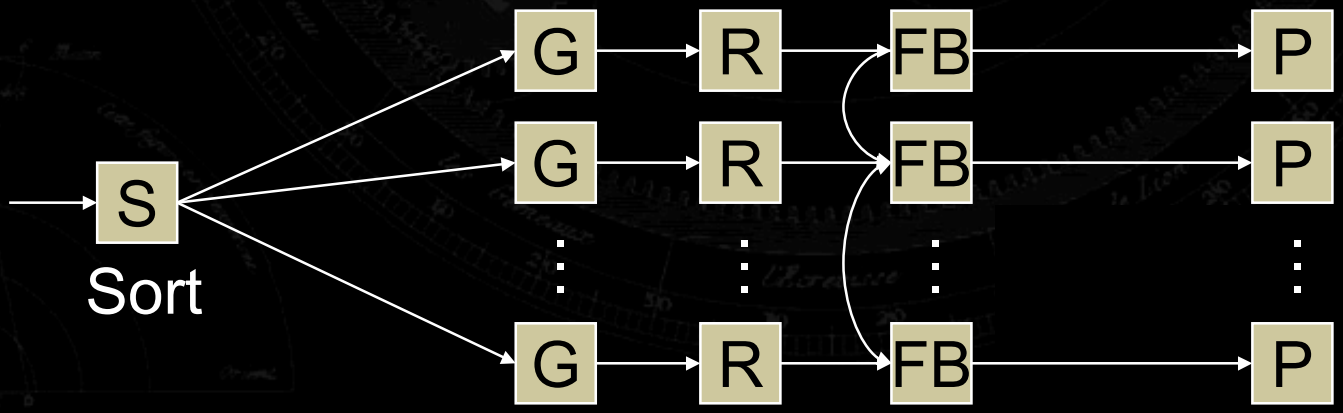
(R. Samanta, J. Zheng, T. Funkhouser, K. Li, JP Singh)

Challenge

- Load balanced with minimal communication requirements

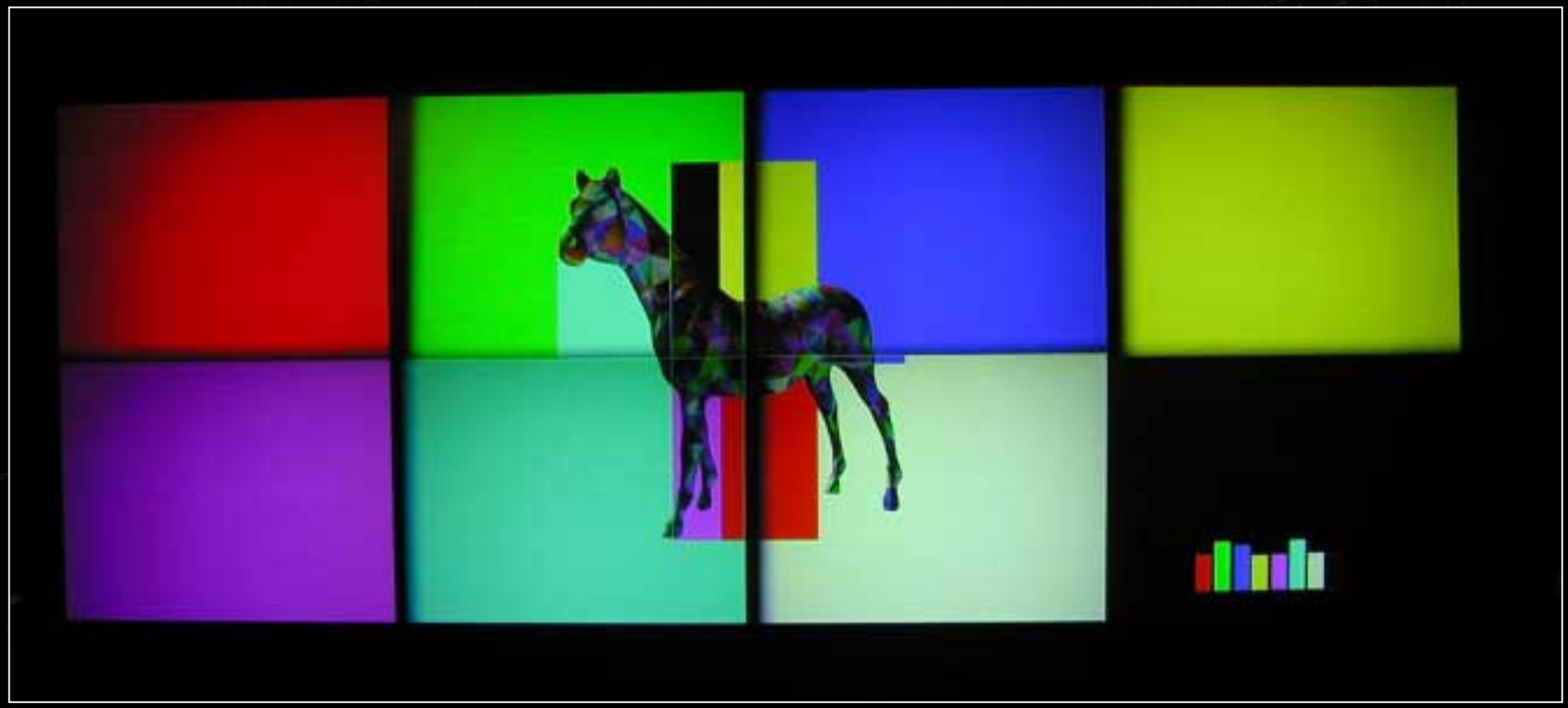
Our approach

- K-D tree space partitioning (sort-first) at object level
- Replicate data base on every node
- Coordinate local and remote rendering pipeline
- Communicate and synchronize using VMMC



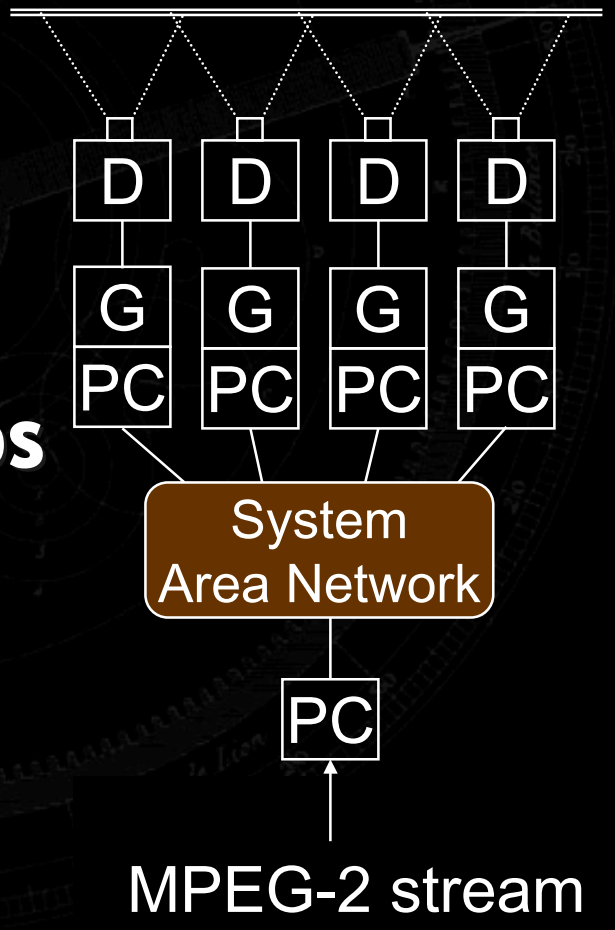
Naïve Rendering without Load-Balancing

K-D Tree Partitioning for Load Balancing



Parallel MPEG-2 Decoding
(H. Chen, K. Li)

- Coordinate PCs
- Minimize communication requirements
- A splitter runs at > 100fps
- A fast MPEG-2 decoder
  - 720p at 50fps on 733Mhz P-III
- Low aggregate bandwidth requirement



MPEG (HDTV resolution)

SIGGRAPH 2001 EXPLORE INTERACTION AND DIGITAL IMAGES

Orion MPEG

Multi-Channel Immersive Sound

Spatializedsound

Coordination

- Coordinate 2 × 8 channels of sound
- 16 speakers (2 subwoofers)

Communication

- Currently communication with a remote client
- Will render spatial sound in parallel

User Interface

Camera-tracked input

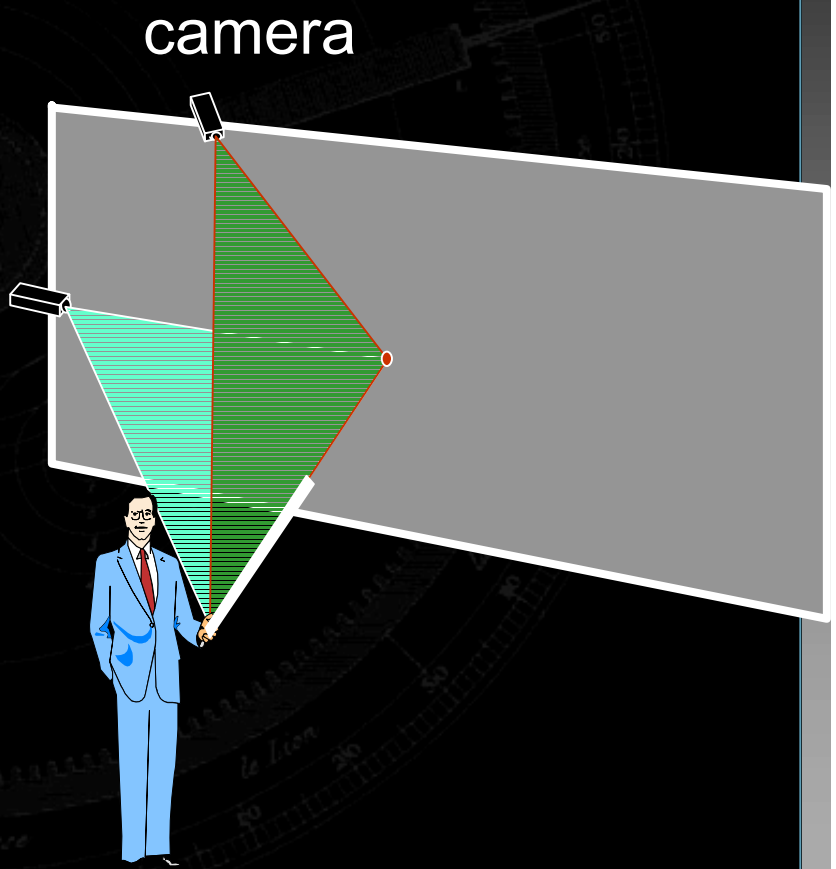
- Triangulation of two video images
- Wand and hand

Voice recognition

- Use commands to drive applications
- Implemented a circuit viewer

Other handheld input

- Gyroscope mouse
- Palm devices



Experiences in Digital Design

Must consider “frameless” design
Allow multiple groups of people to view
Life size makes a big difference
No need to rapidly change images
Font sizes can range from 2 to 600 points
Spatial sound is important for story-telling

Early Experience Summary

Build a fun toy, people will play with it

It is possible to build an inexpensive, scalable display wall system

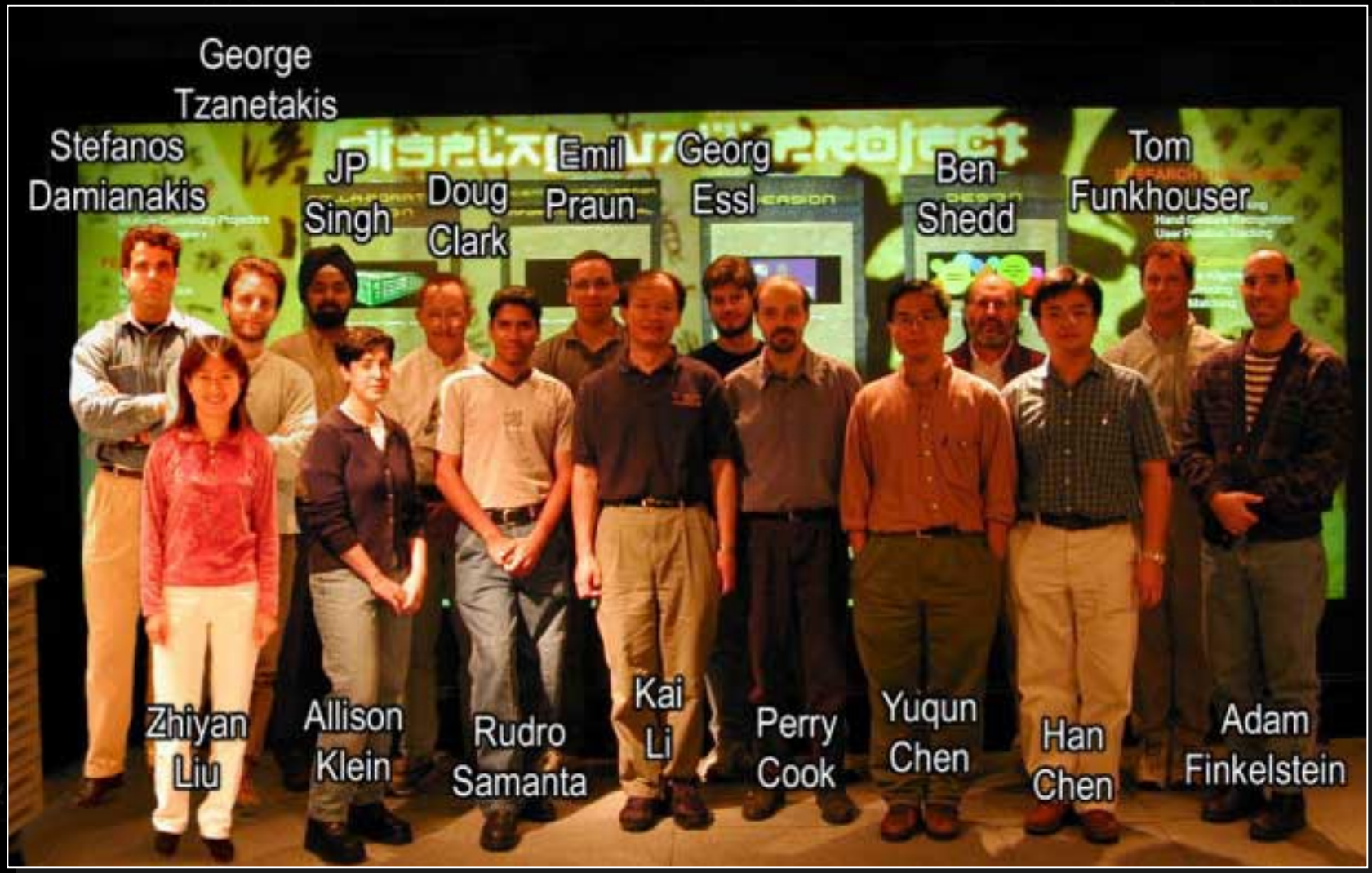
- Overcome commodity components' features

Wall-size display systems will lead us to rethink software and content creation

- Algorithms that trade space for less communication
- Content design
- Multiple viewers

This is just a beginning

Princeton Display Wall Team



Scalable multi-user display surfaces with blending and compositing

Alan Heirich, Compaq
Santiago Lombeyda, Caltech

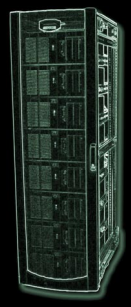
with contributions from Laurent Moll, Mark Shand, Dave Garcia, Bob Horst (Compaq) and Ravi Ramamoorthi (Stanford)

SIGGRAPH 2001
EXPLORE INTERACTION AND DIGITAL IMAGES
COMPAQ

Scalable display surfaces

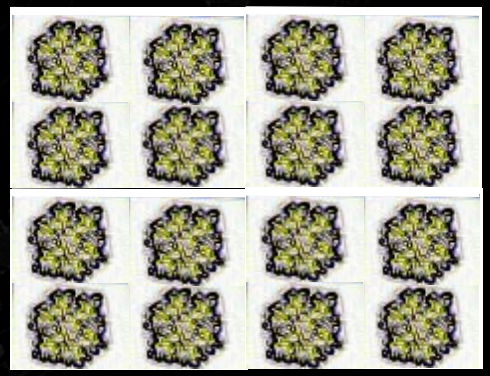
Rendering computers
1...1024...

Scalable display surfaces

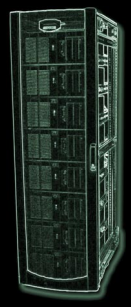


Rendering computers
1...1024...

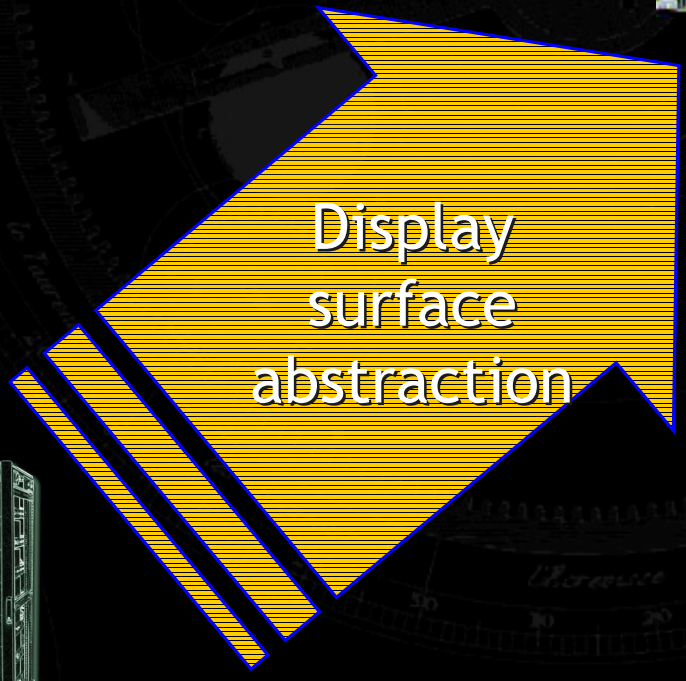
Display panels 1…512...



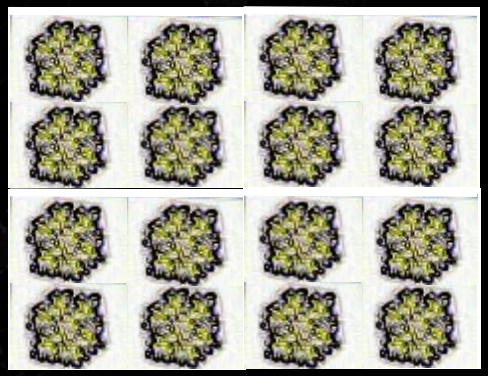
Scalable display surfaces



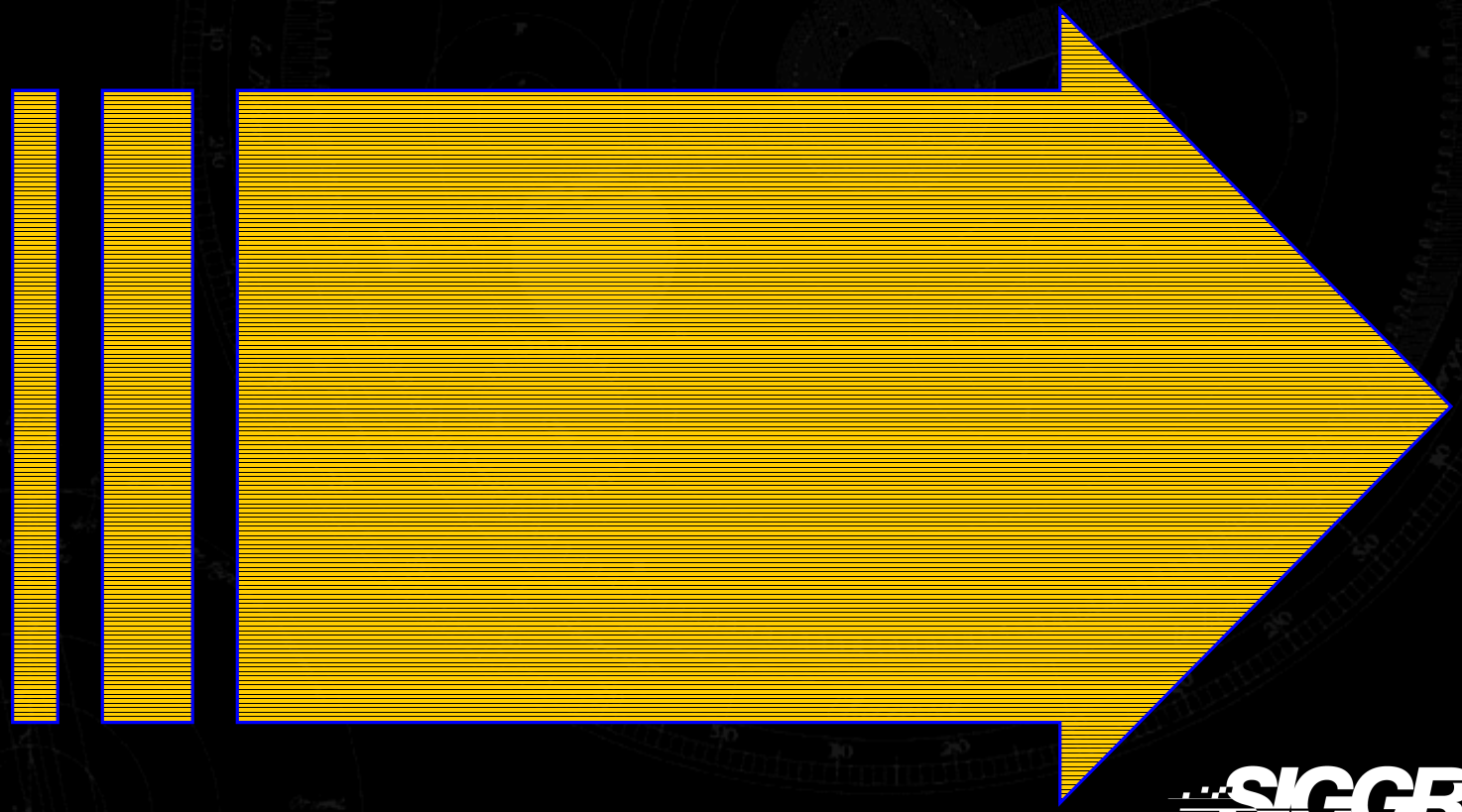
Rendering computers
1...1024...



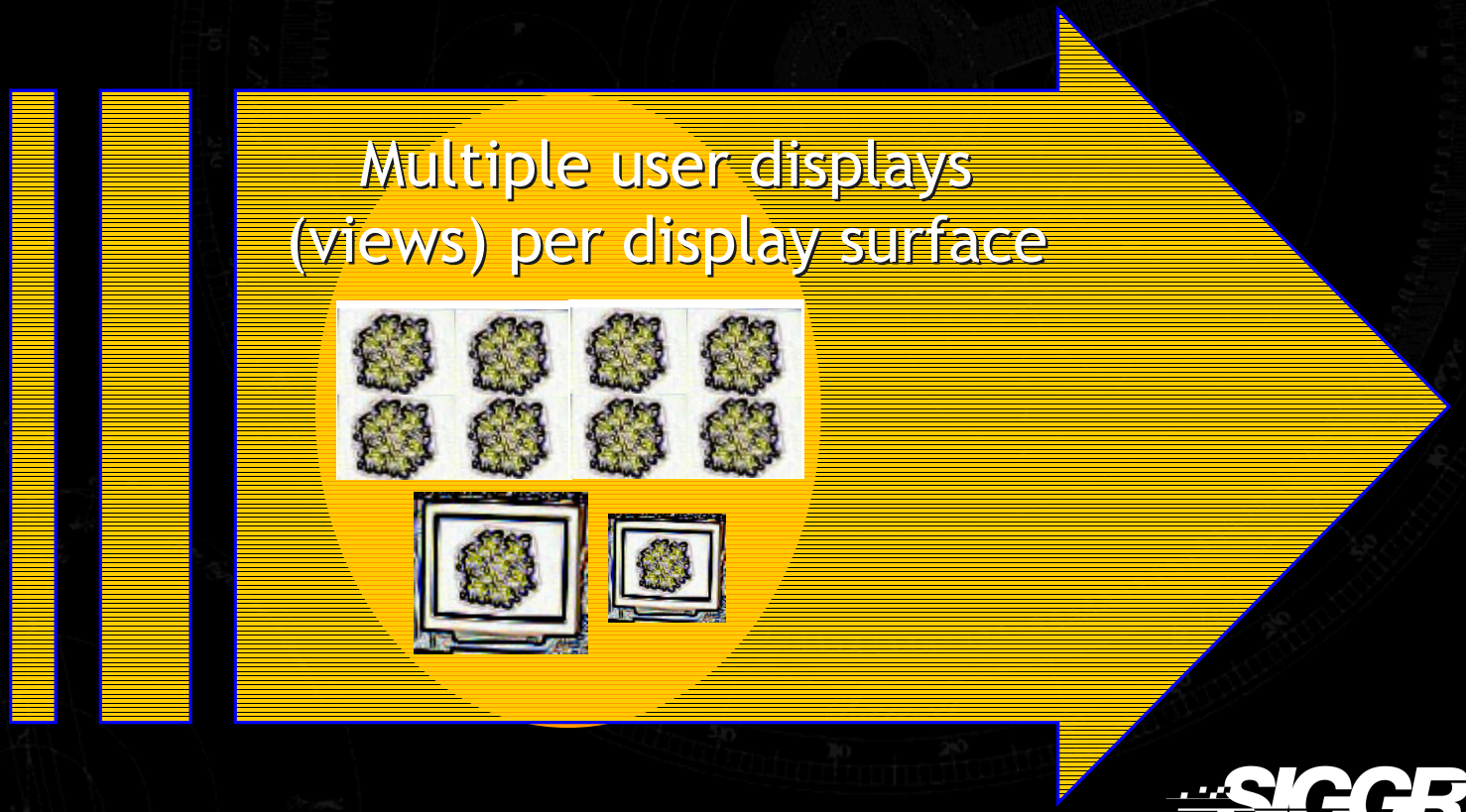
Display panels 1…512...



Display surface abstraction



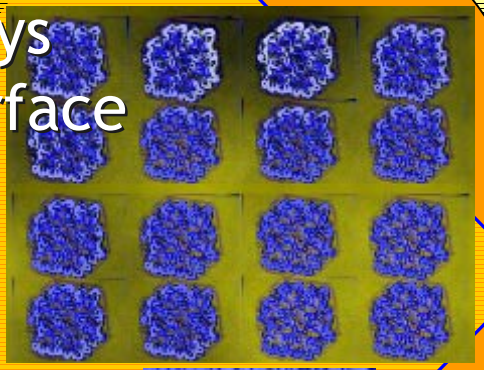
Display surface abstraction



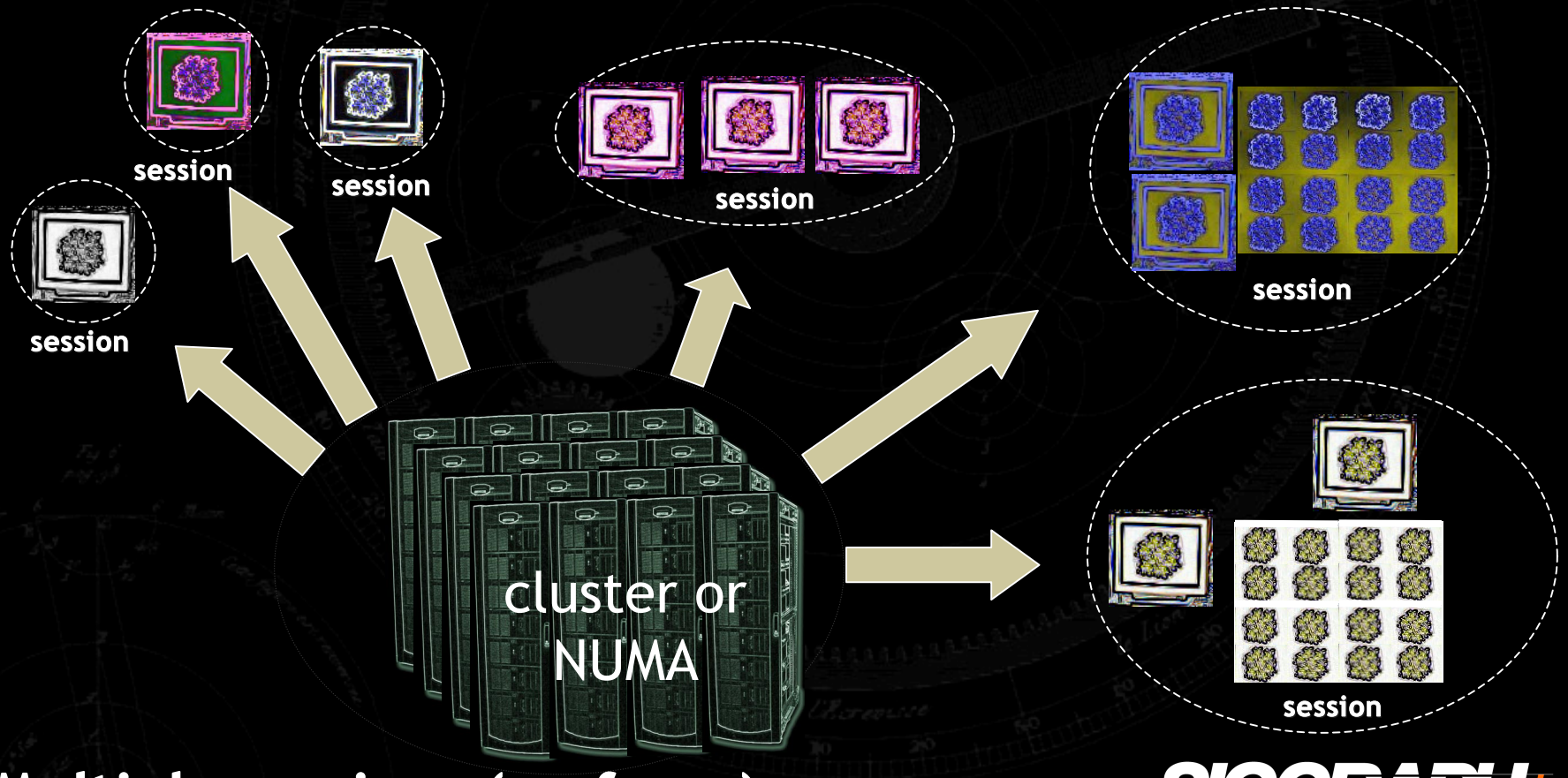
Display surface abstraction

Multiple display surfaces per supercomputer

Multiple user displays (views) per display surface



Multi-user visualization environment



Multiple sessions (surfaces)
Multiple users (views) per session

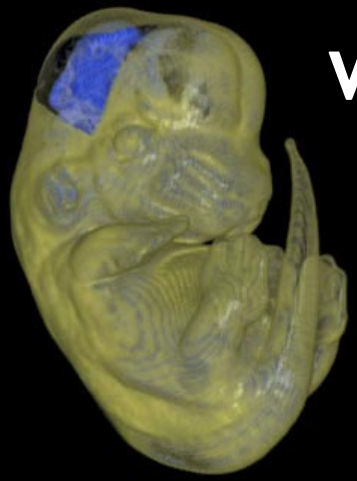
Application requirements

- Scalable display surface
- Independent viewports
  - dynamic rescaling
- Extensible compositing
- Blending & Porter-Duff operators
  - blending = compositing + reordering
- Interactive end-to-end latency

Extensible compositing

Depth compositing
visual simulators
isosurface rendering

Extensible compositing

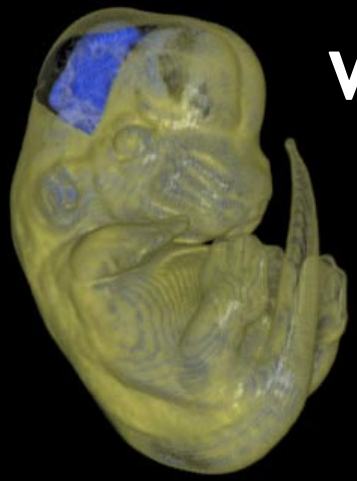


Volumetric blending
non-commutative
Porter-Duff operators

Santiago Lombeyda,
Russ Jacob &
Scott Fraser

Depth compositing
visual simulators
isosurface rendering

Extensible compositing



Volumetric blending
non-commutative
Porter-Duff operators

Santiago Lombeyda,
Russ Jacob &
Scott Fraser

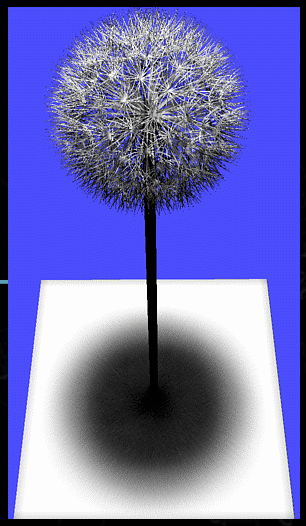


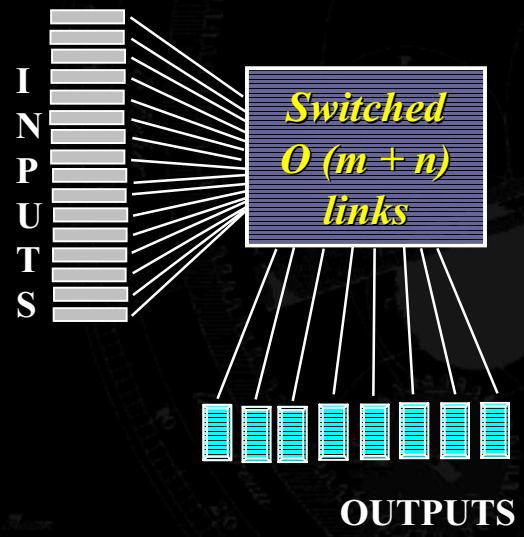
Photo-realistic
image projections

SIGGRAPH 2000
Agrawala, Ramamoorthi,
Heirich & Moll

Depth compositing
visual simulators
isosurface rendering

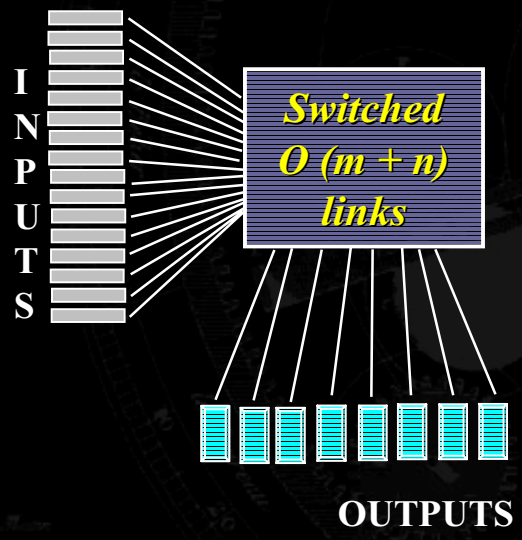
Others ...

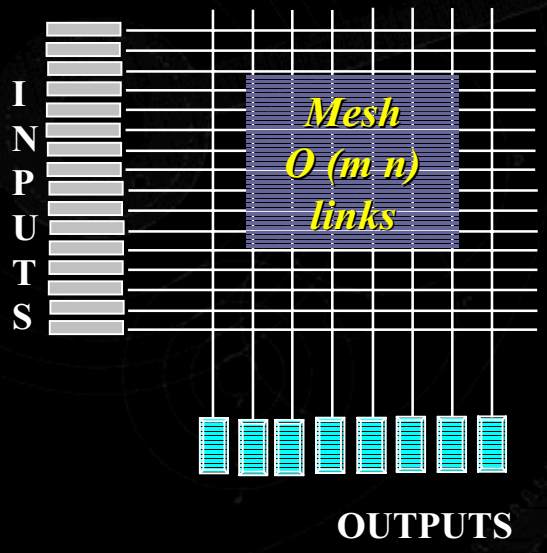
To switch or to mesh?



cost = O (links)

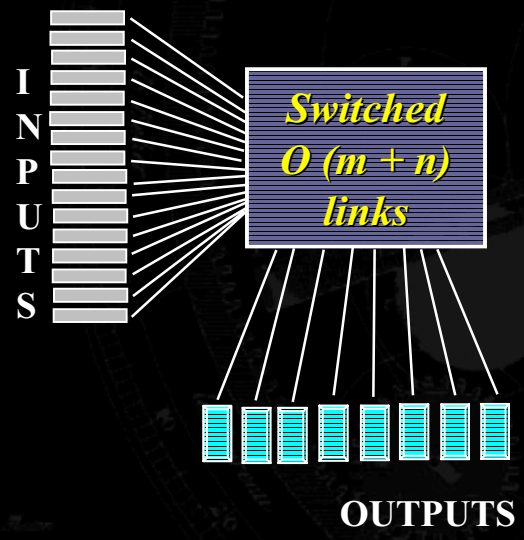
To switch or to mesh?

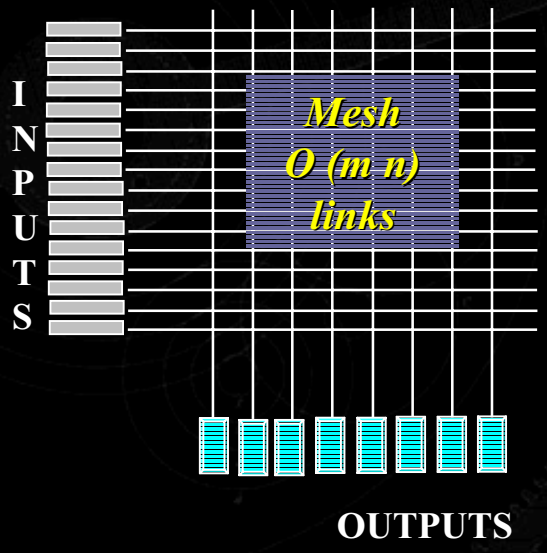




cost = O (links)

To switch or to mesh?





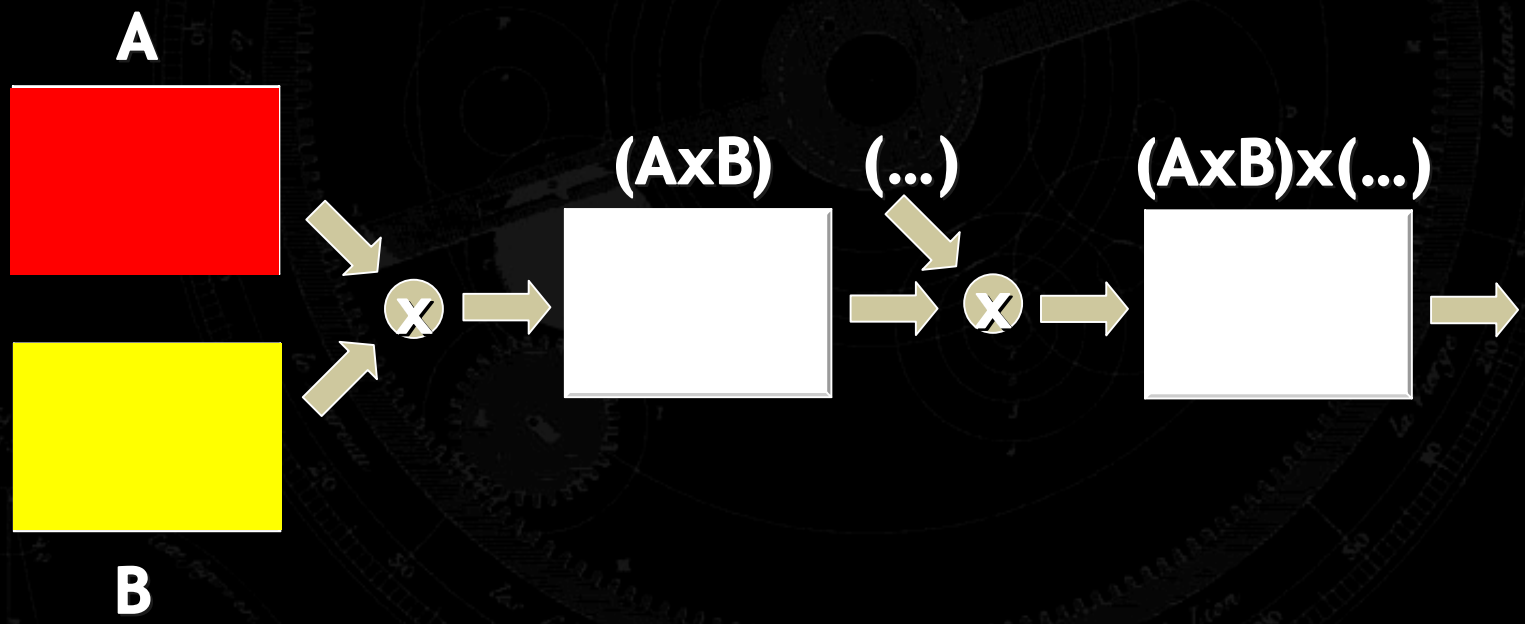
cost = O (links)

m,n	mesh (mn)	switched (m+n)
128,32	4096	160
1024,128	131,072	1152

A dynamic mapping problem

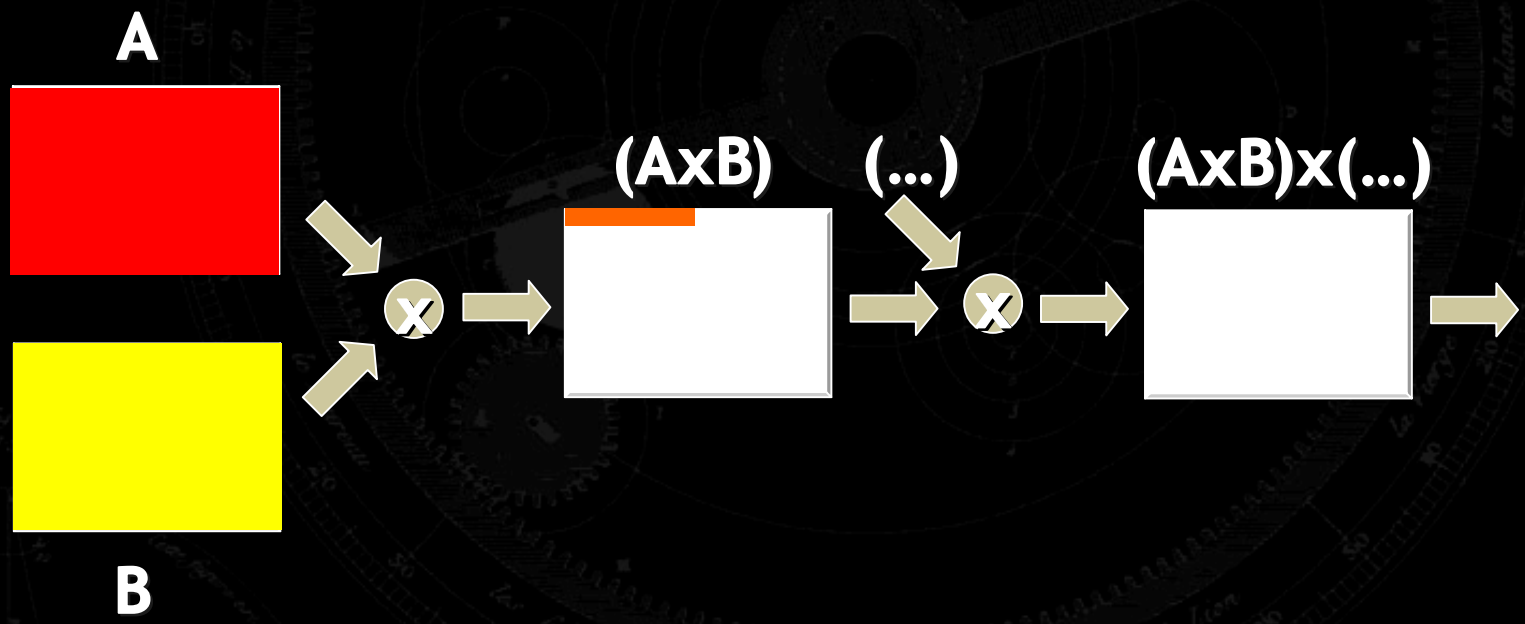
- **Physical topology != logical topology**
  - except depth compositing (commutative operator)
  - defines a *graph embedding* problem
  - intrinsic to every distributed computation
- **Logical topology is a compositing pipeline**
  - embed pipeline graph into physical network/switch
  - need to guarantee contention-free routing
- **Dynamic mapping = Hamiltonian circuit**
  - solution for Clos networks (e.g. Myrinet, Quadrics)
  - published PVG2001
    `http://www.gg.caltech.edu/pvg2001`

Pipeline abstraction



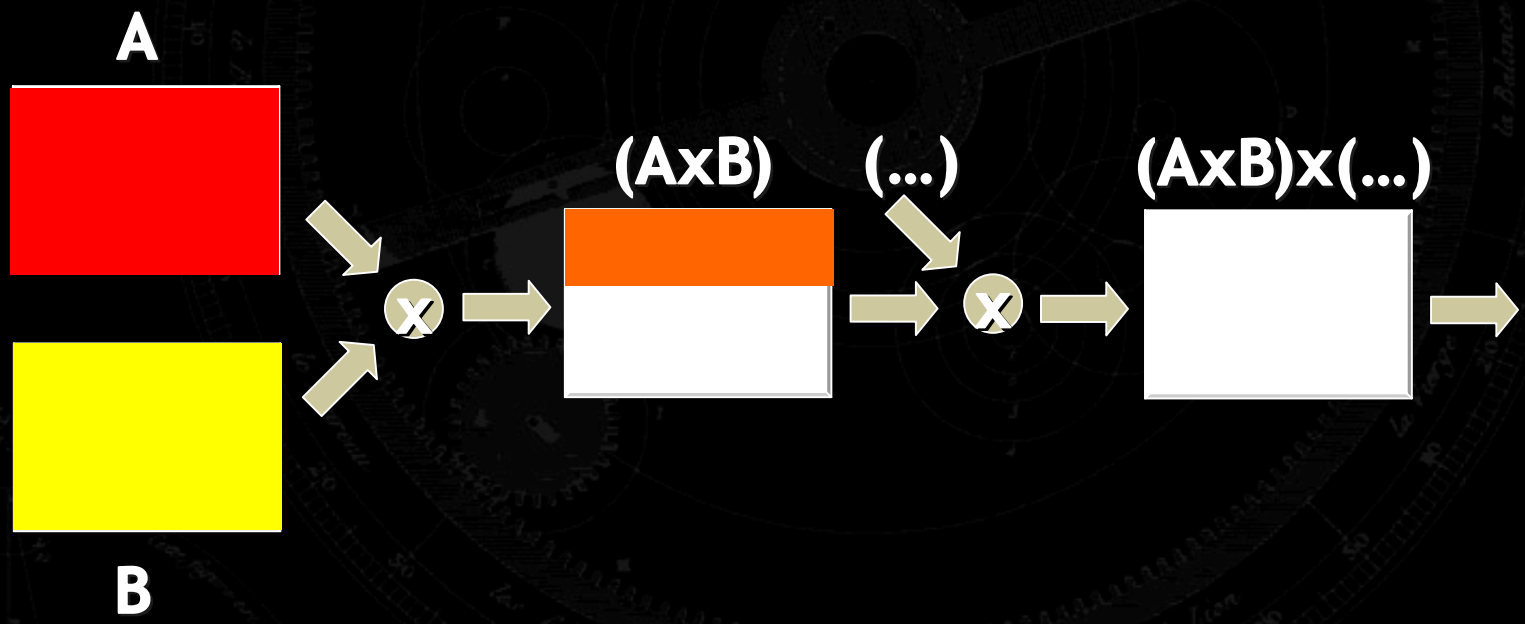
high latency
sequentialcompositing

Pipeline abstraction



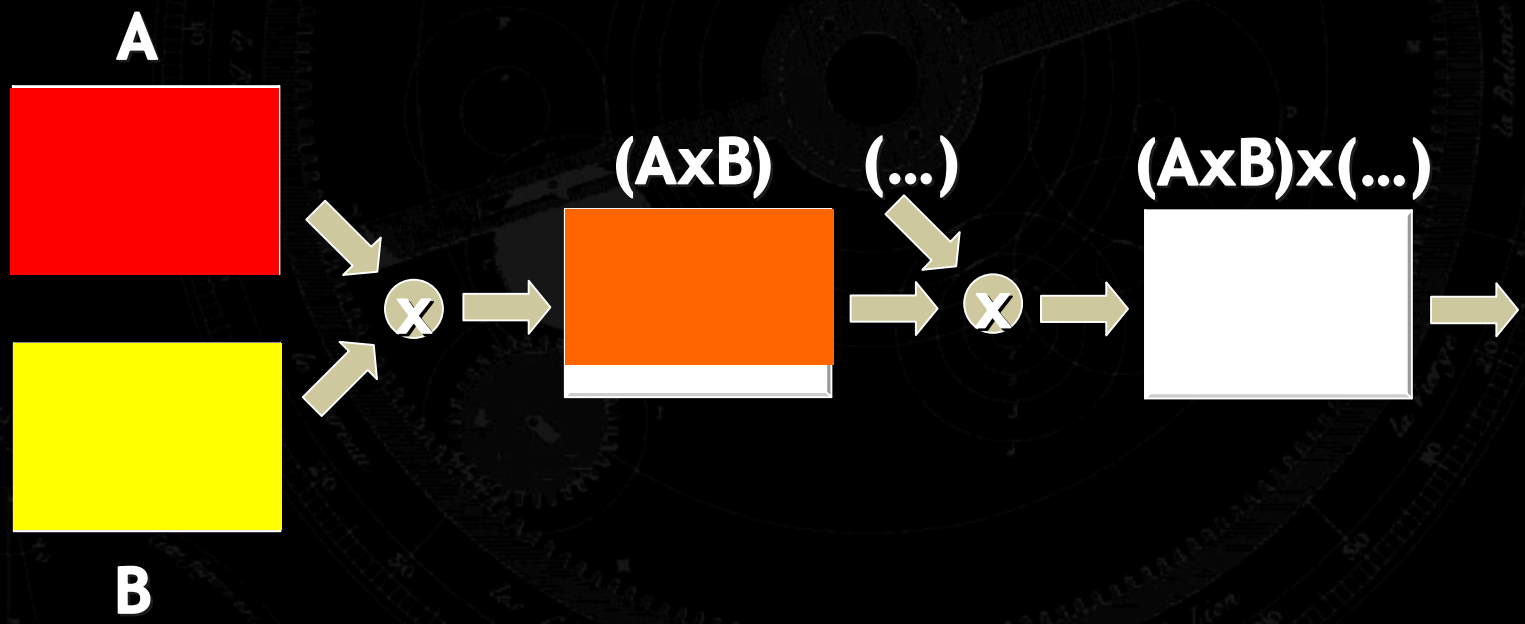
high latency
sequentialcompositing

Pipeline abstraction



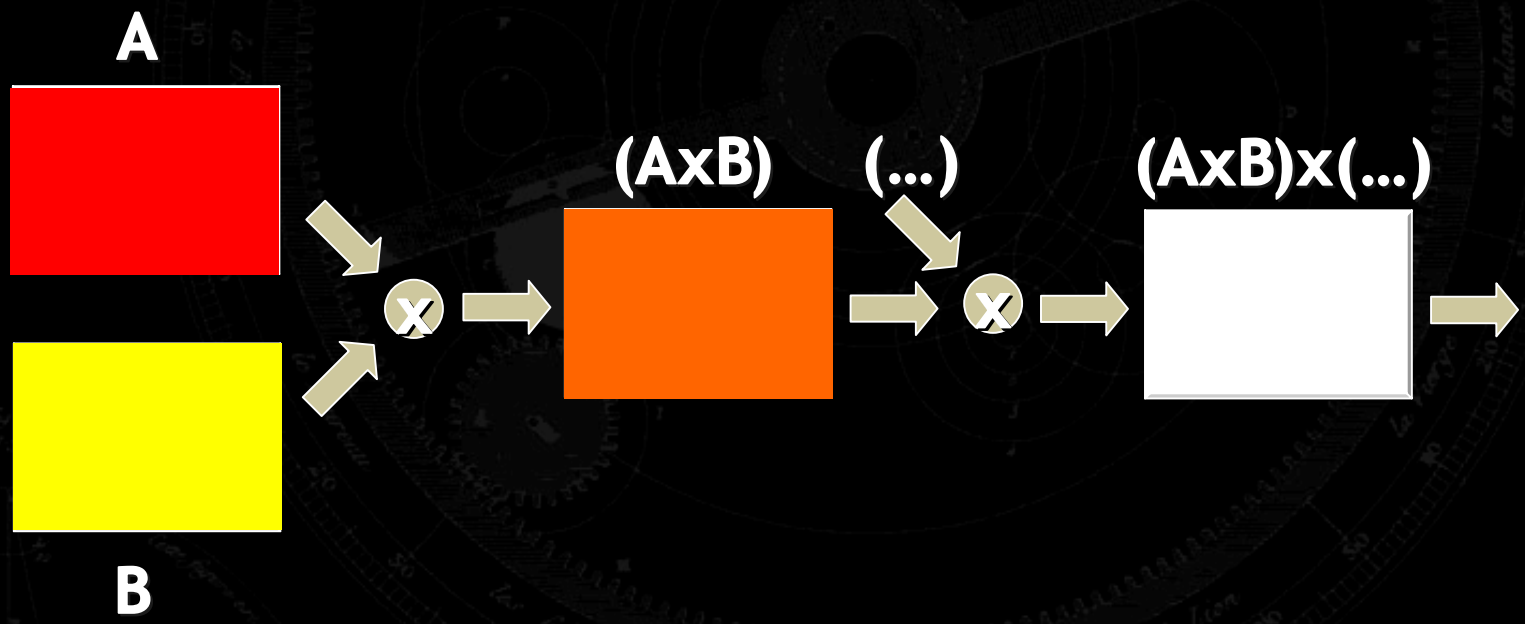
high latency
sequentialcompositing

Pipeline abstraction



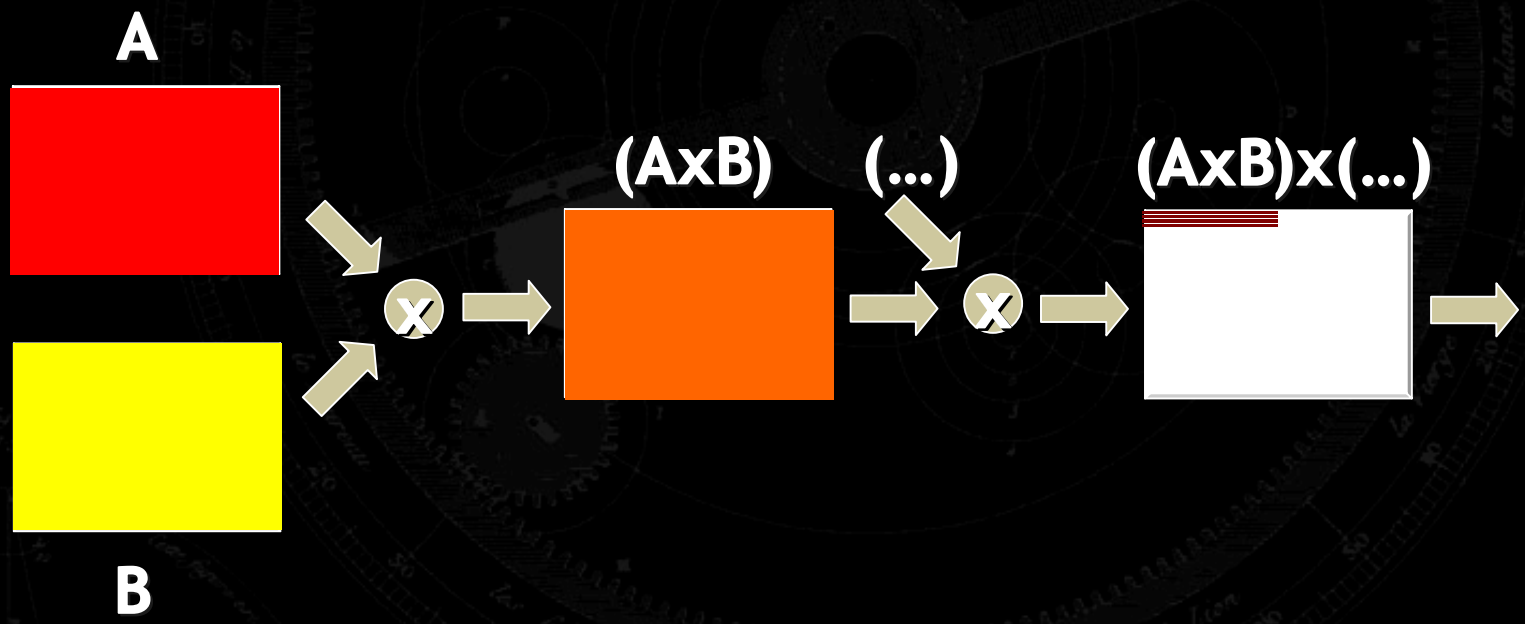
high latency
sequentialcompositing

Pipeline abstraction



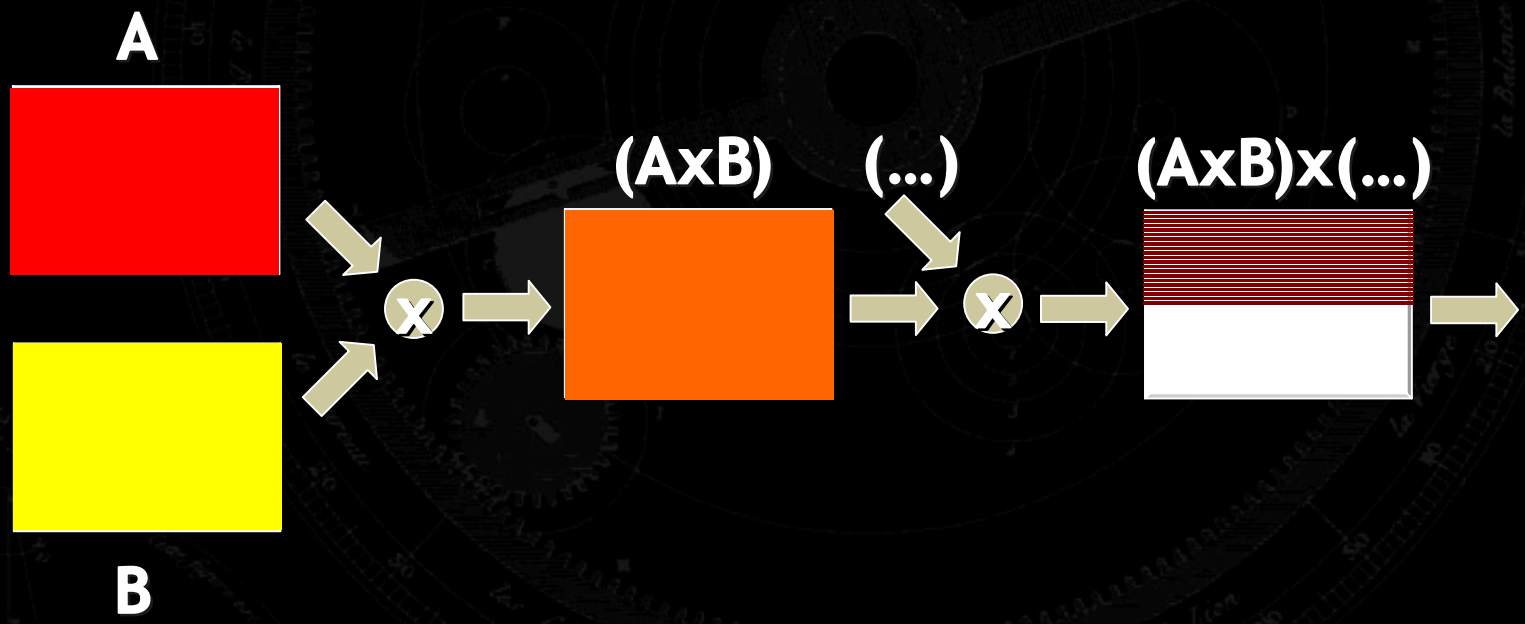
high latency
sequentialcompositing

Pipeline abstraction



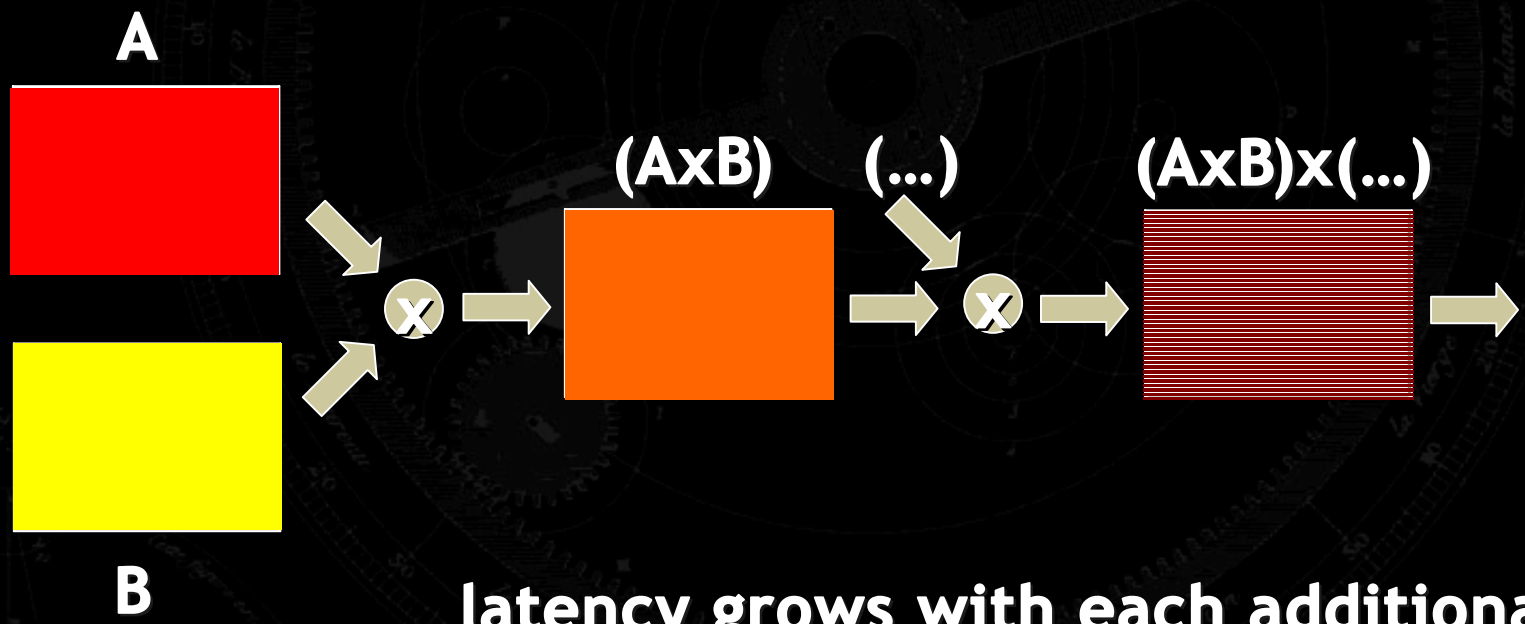
high latency
sequentialcompositing

Pipeline abstraction



high latency
sequentialcompositing

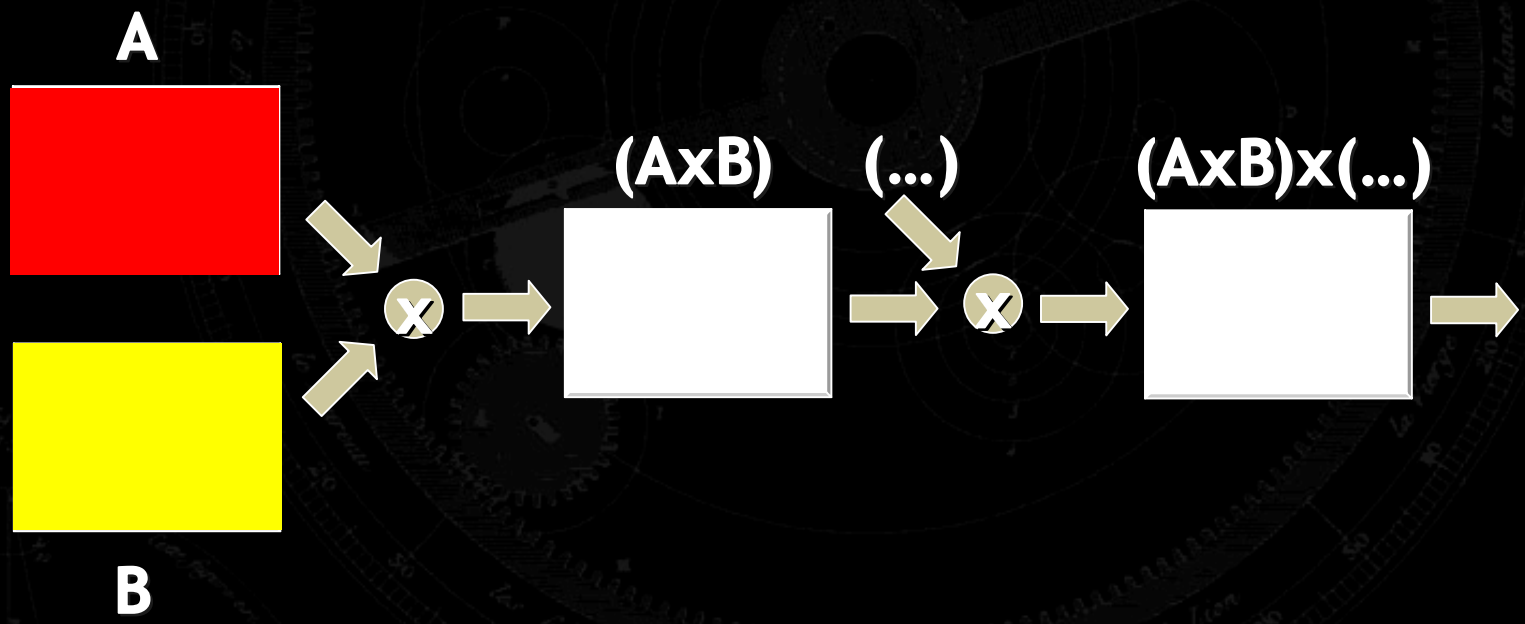
Pipeline abstraction



latency grows with each additional stage

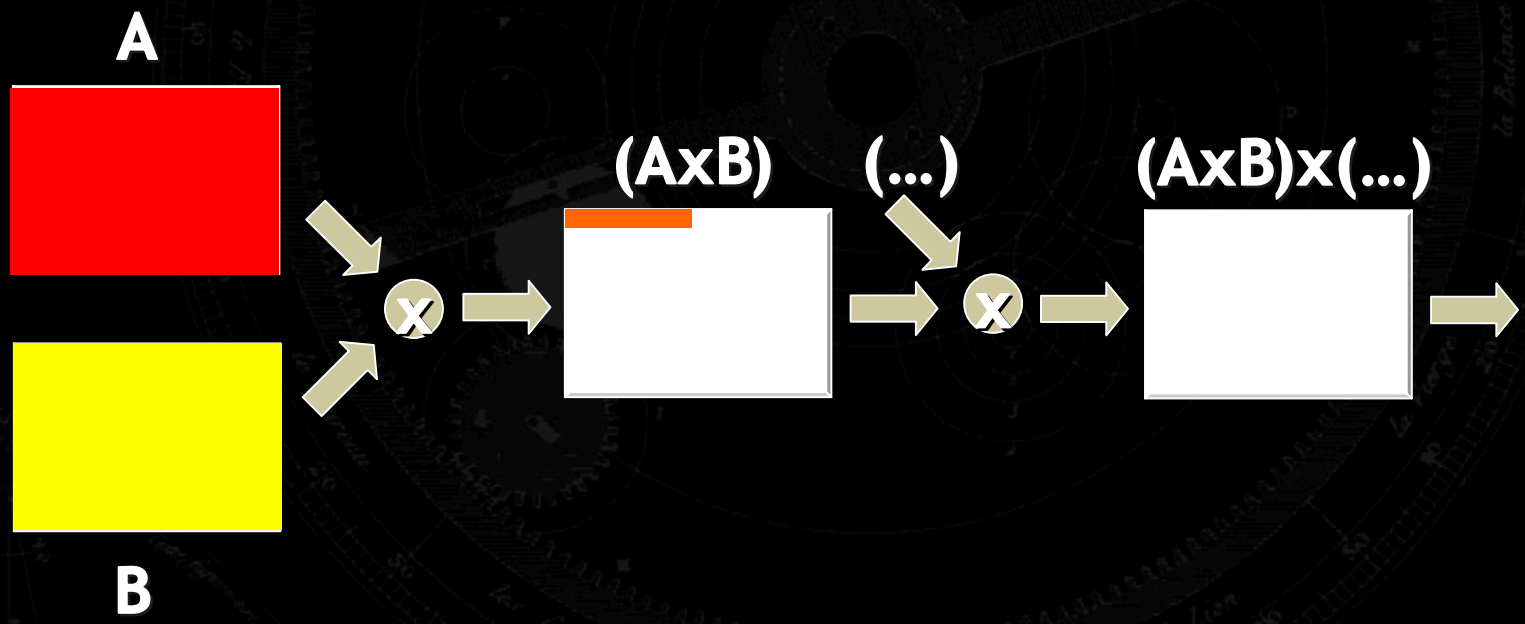
high latency
sequentialcompositing

Pipeline abstraction



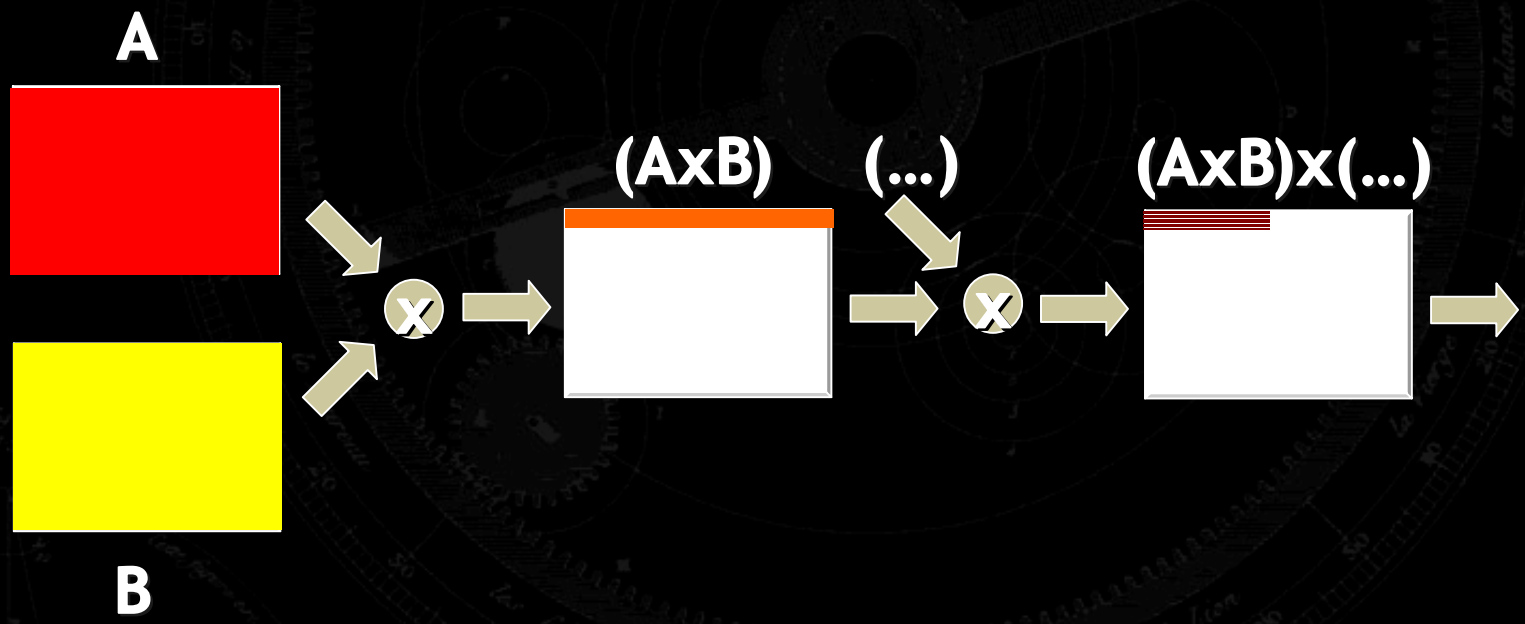
interactive latency
fine grainedcompositing

Pipeline abstraction



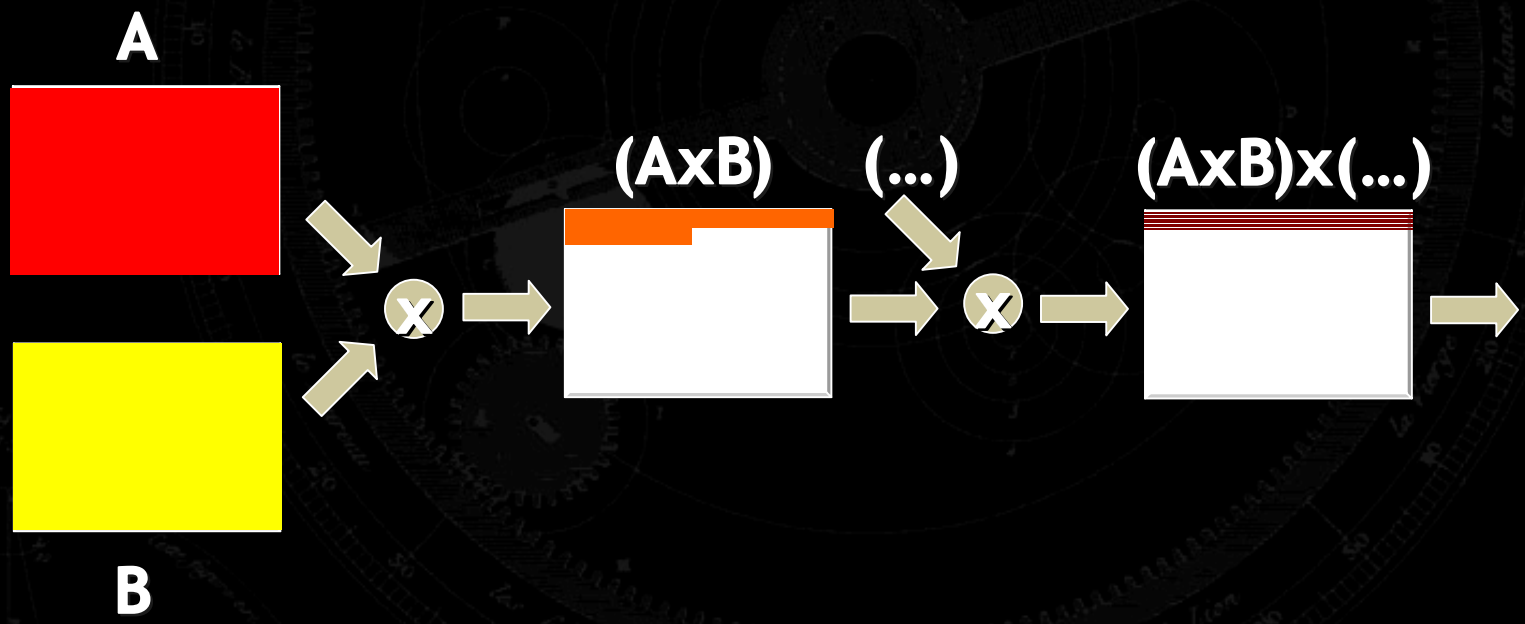
interactive latency
fine grainedcompositing

Pipeline abstraction



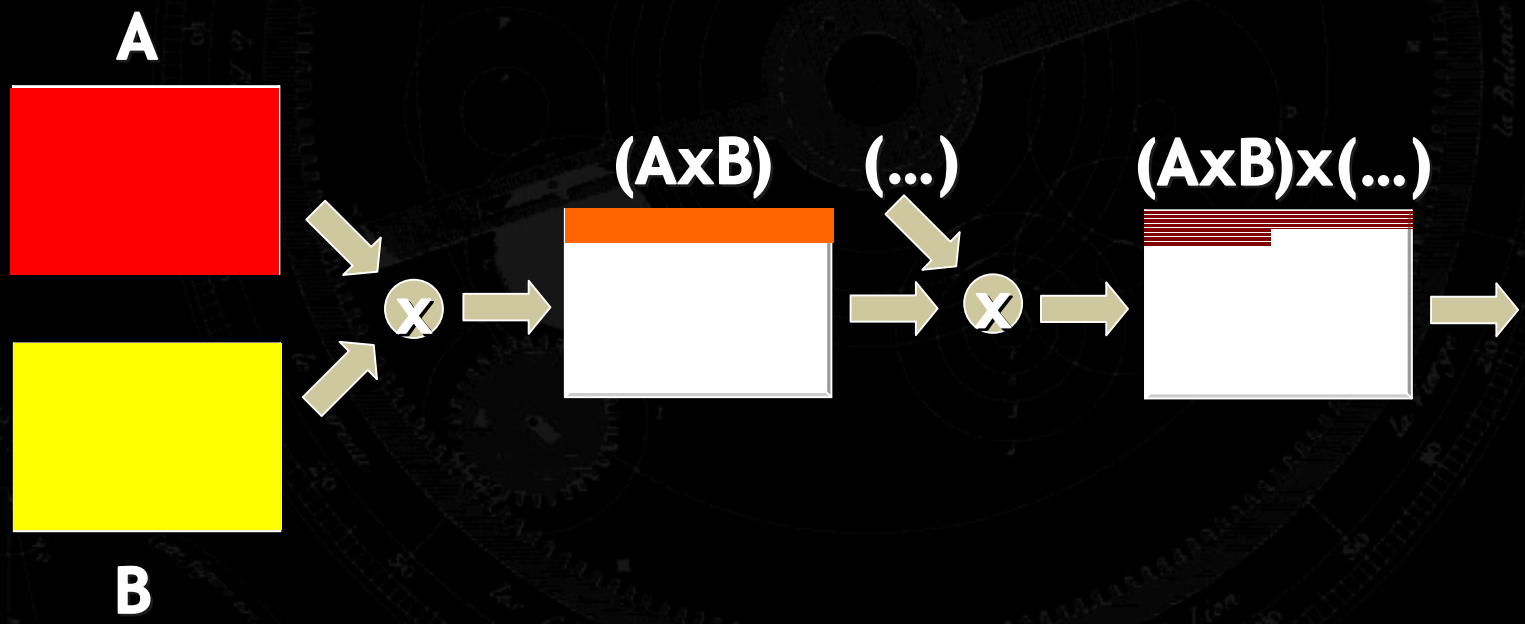
interactive latency
fine grainedcompositing

Pipeline abstraction



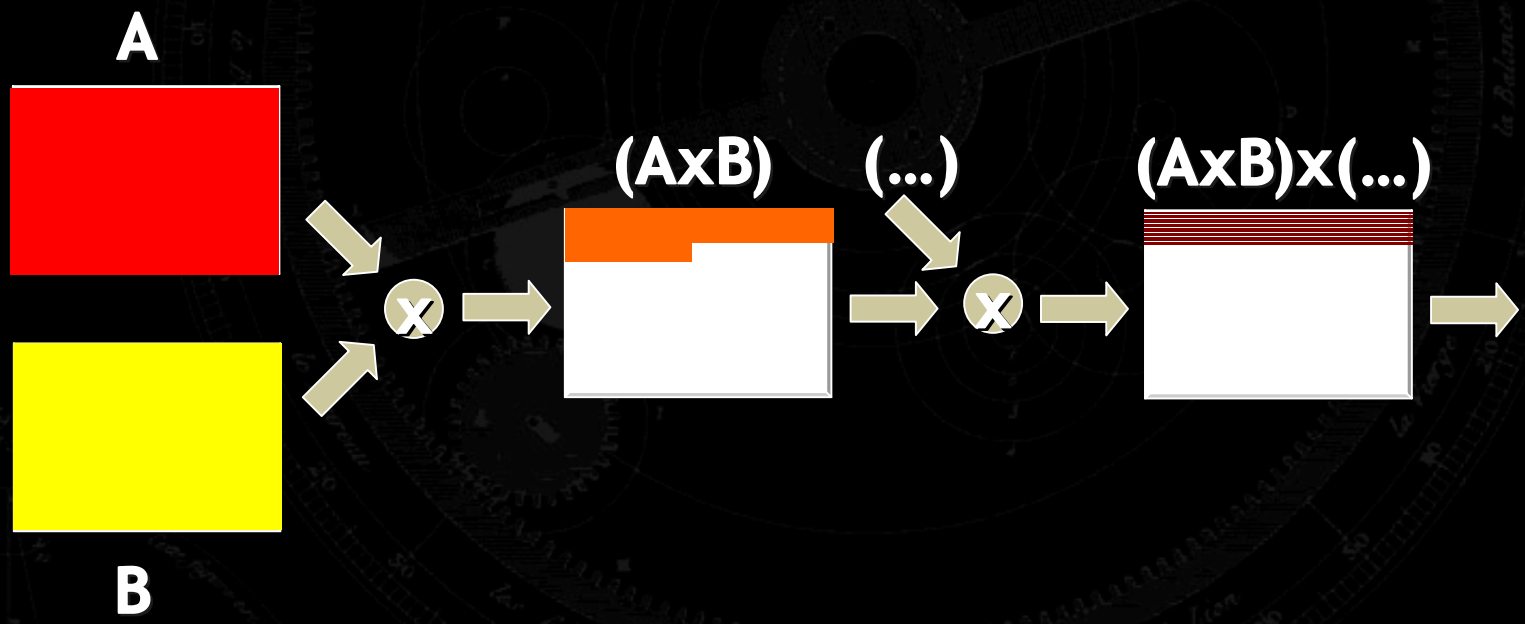
interactive latency
fine grainedcompositing

Pipeline abstraction



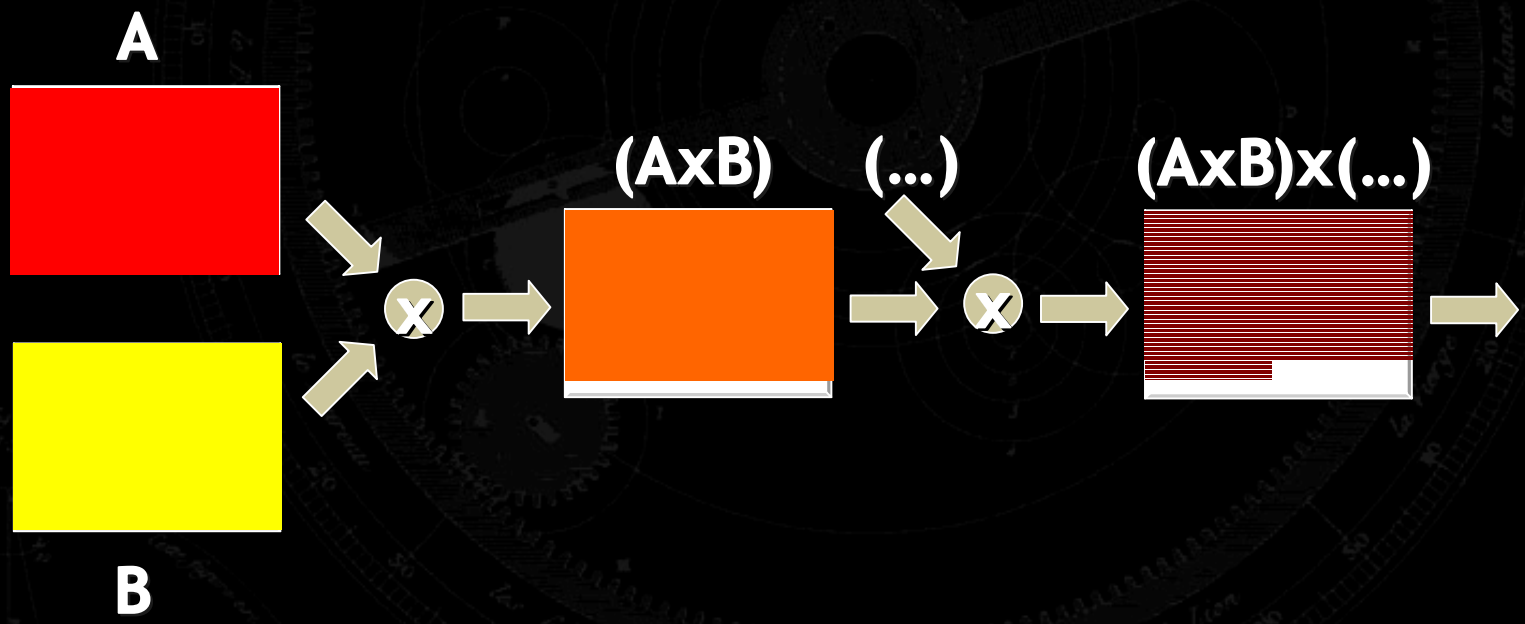
interactive latency
fine grainedcompositing

Pipeline abstraction



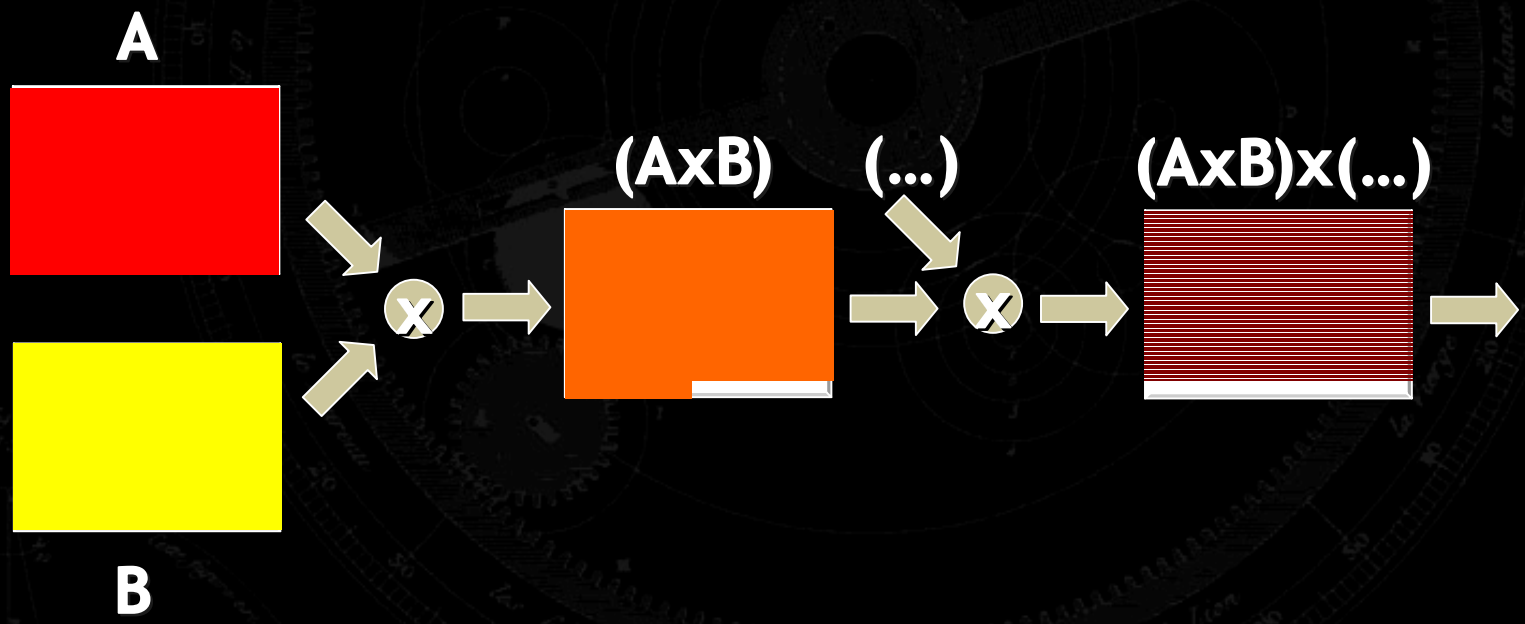
interactive latency
fine grainedcompositing

Pipeline abstraction



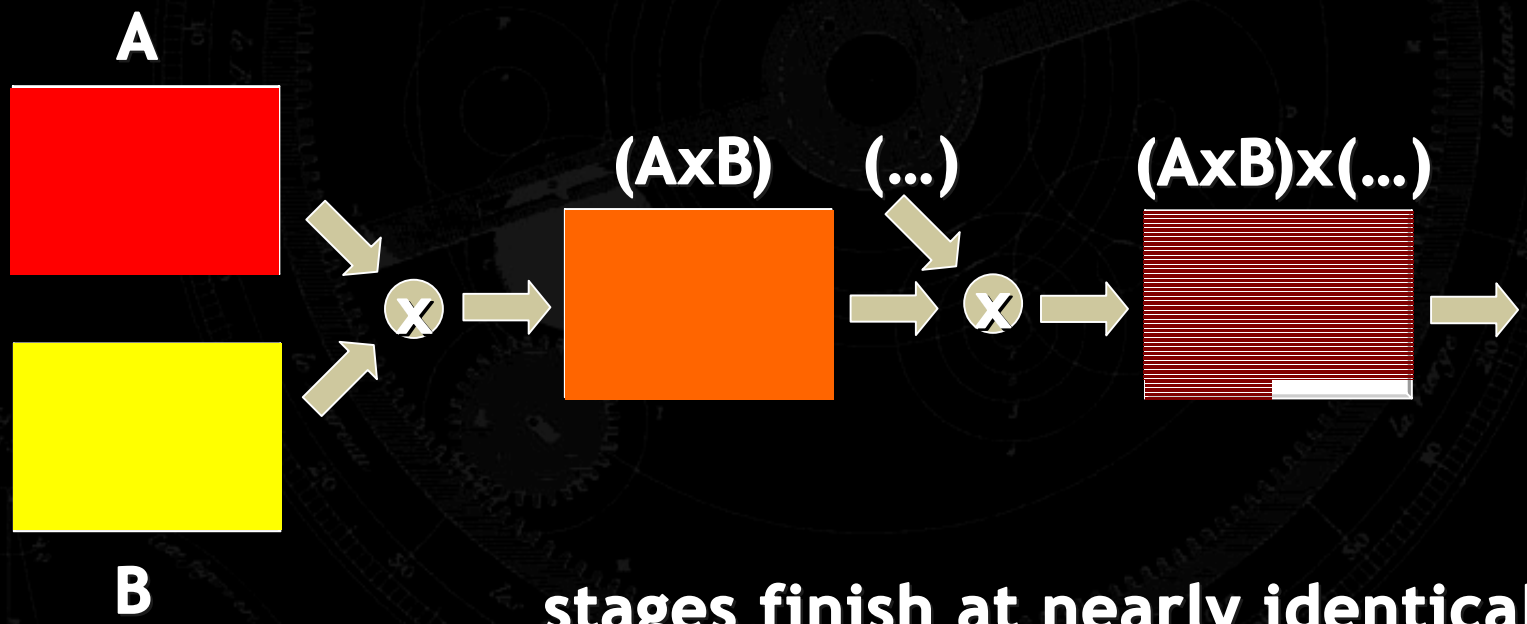
interactive latency
fine grainedcompositing

Pipeline abstraction



interactive latency
fine grainedcompositing

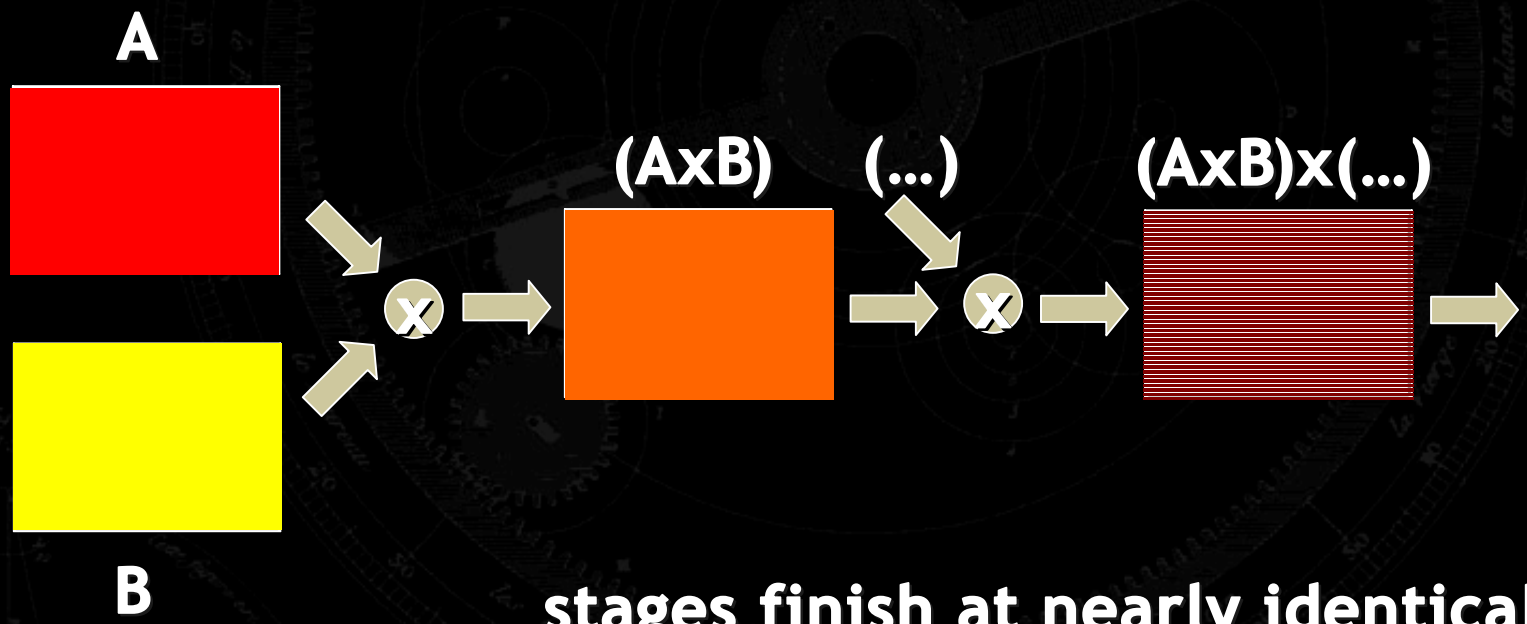
Pipeline abstraction



stages finish at nearly identical times

interactive latency
fine grainedcompositing

Pipeline abstraction

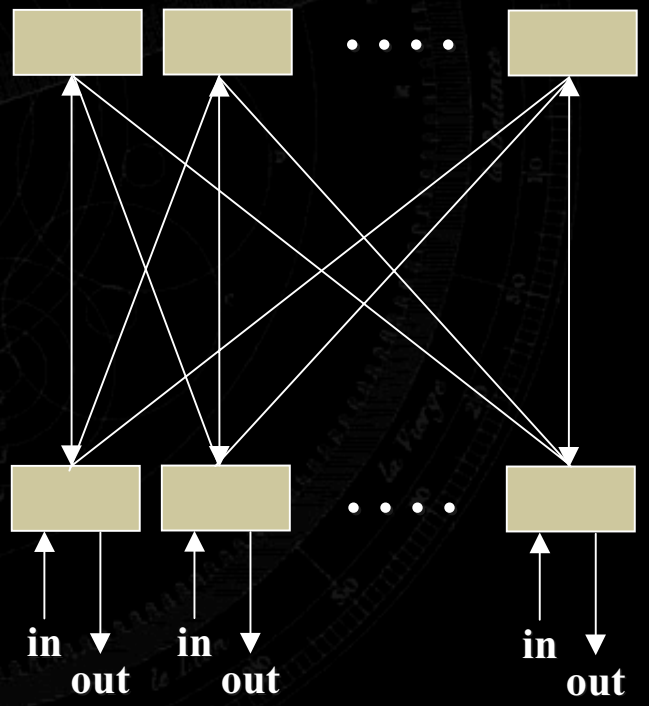


stages finish at nearly identical times

interactive latency
fine grainedcompositing

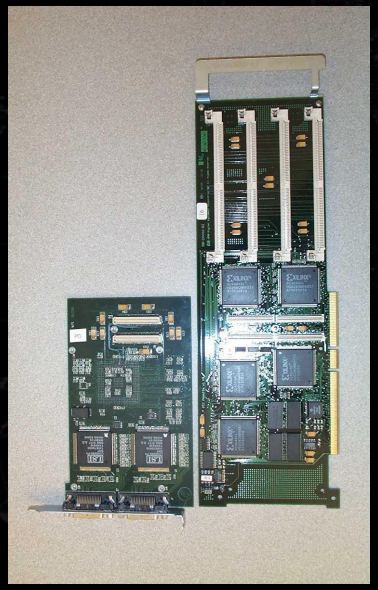
Dynamic mapping solution

- Symmetric Clos networks (full duplex)
- Hamiltonian circuit embedding problem
- Forward mapping by network address
- Guarantee routability for any pipeline order
- Constructive proof, PVG2001
  (Lombeyda, Shand, Moll, Breen & Heirich)



Hardware prototypes

Sepia-2 (2000)
360 MB/s sustained
ServerNet-II
1280x1024 RGBA 36 Hz



Sepia-1 (1998)
80 MB/s sustained
ServerNet-I
640x480 RGBA 34 Hz

Sepia-3 (2002)

> 1 GB/s sustained

Infiniband
HyperTransport

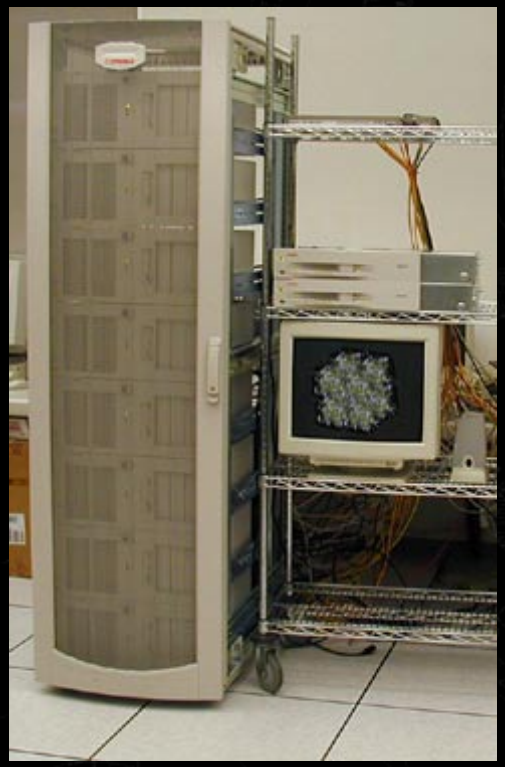
2048x2048 RGBA 80 Hz

new features

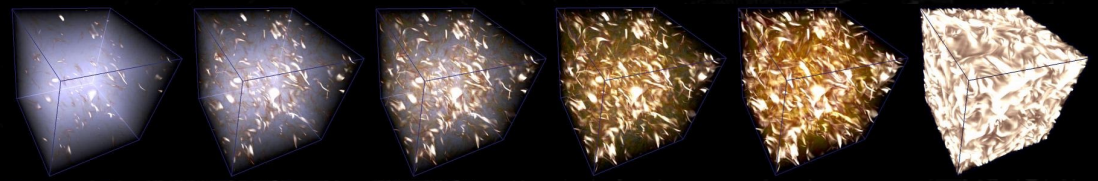
Untested solutions - Sepia-3

- DVI image acquisition
- Viewport independence
  - span multiple display panels
  - rescalable in real time
- Soft shadow mapping - display phase
  - SIGGRAPH 2000 paper (Agrawala et al)

Scalable volumetric rendering



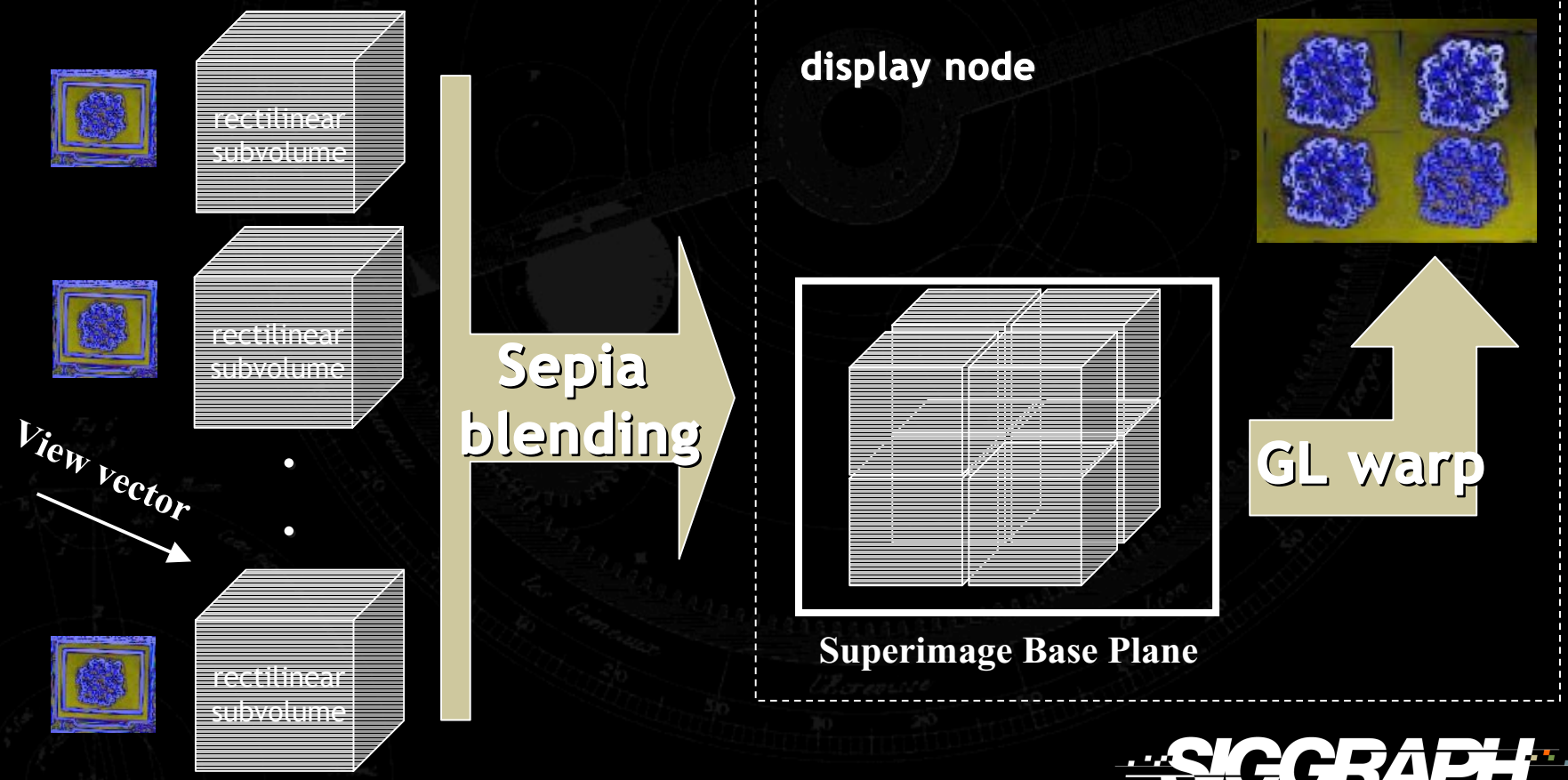
NSF TeraVoxelproject
Kilo-Frame/Sec camera
ASCI Center for Simulation of
Dynamic Response of Materials



Ravi Samtaney (simulation), Santiago Lombeyda (visualization)

- Hardest compositing case
  - requires dynamic remapping
- Real application
- Small scale demonstration
  - theory extends to large networks

Concurrent shear-warp ray casting



Chromium: An Open-source Cluster Rendering System

Greg Humphreys

Stanford University

SIGGRAPH 2001
EXPLORE INTERACTION AND DIGITAL IMAGES

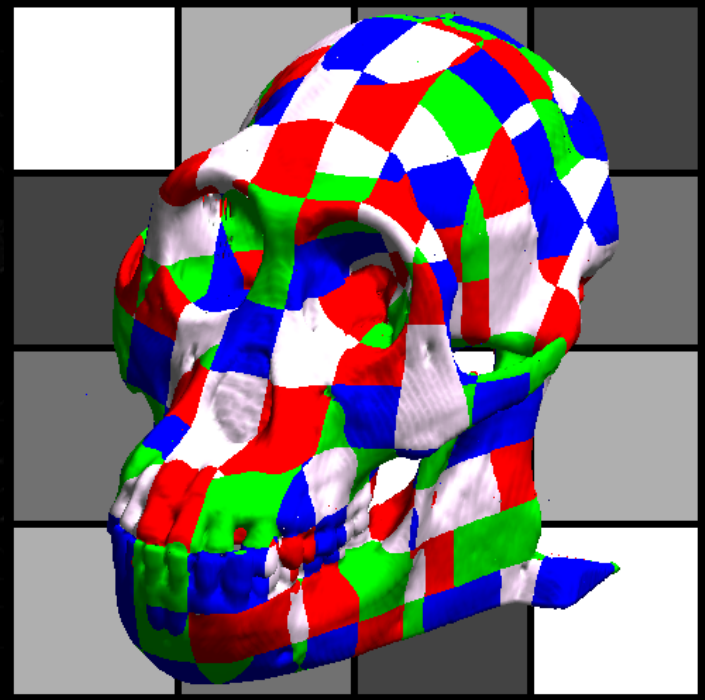
Motivation

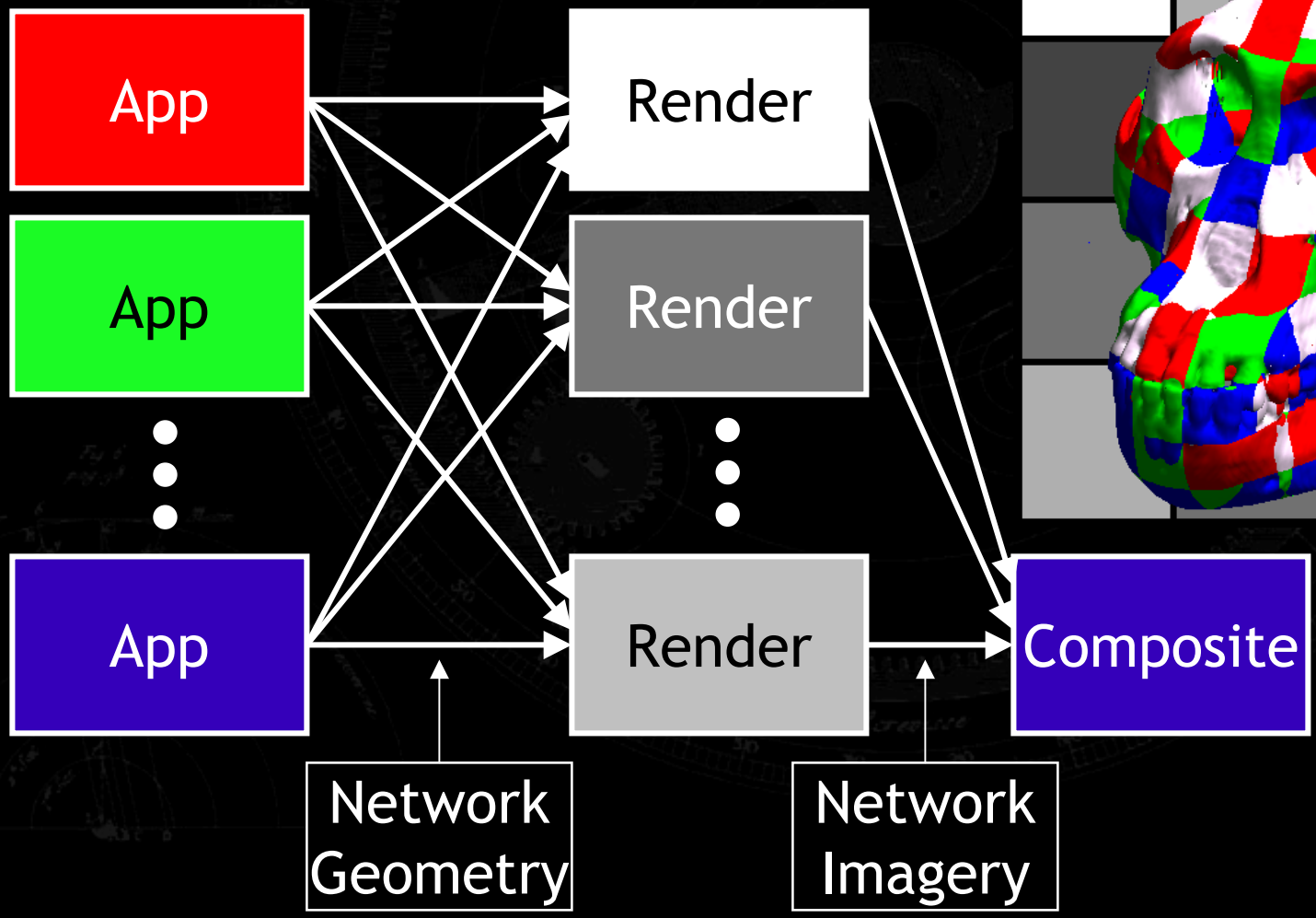
WireGL's power derives from its flexibility

- Any OpenGL program, plus a parallel interface
- Arbitrary assignment of tiles to servers
- Independently set number of clients and renderers
- Good scalability for different application types

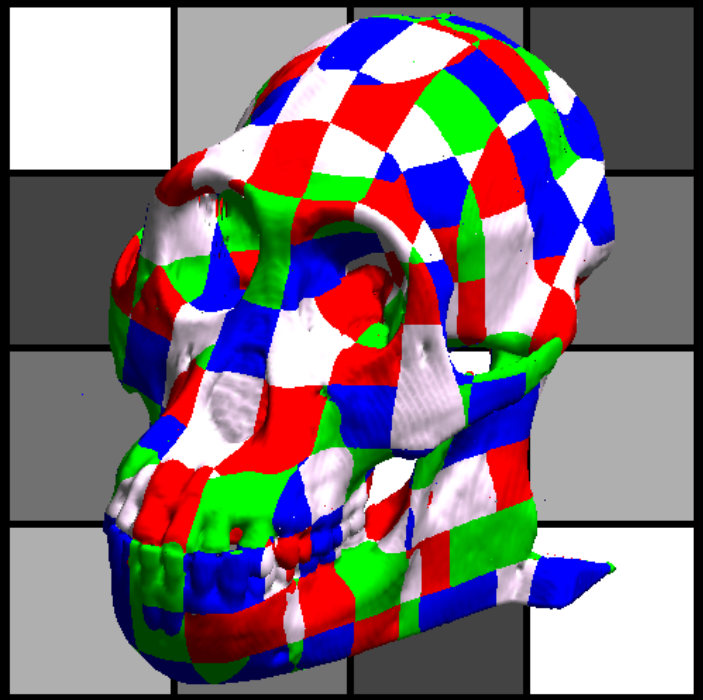
But WireGL has some undesirable restrictions

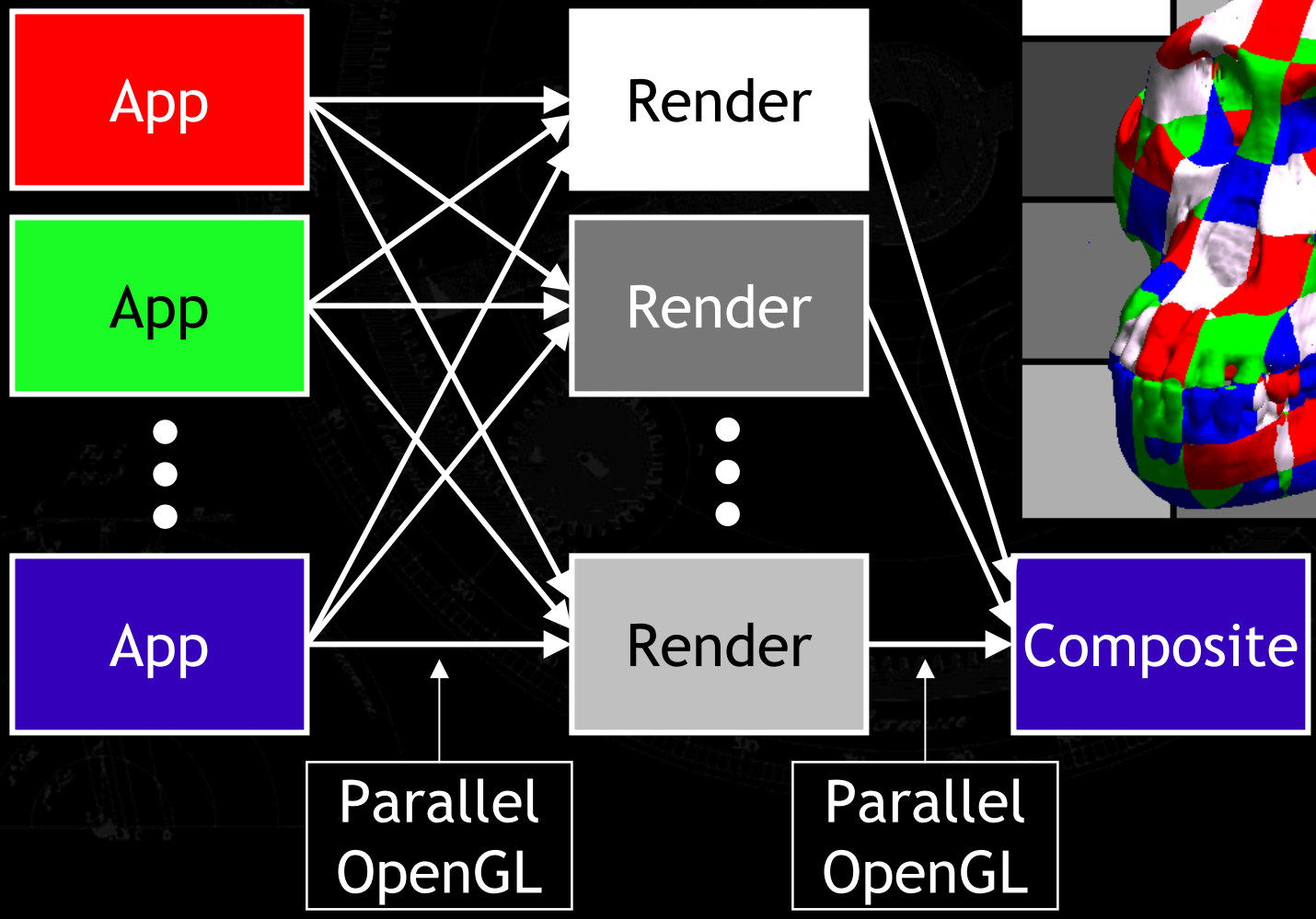
WireGL "Visualization Server"





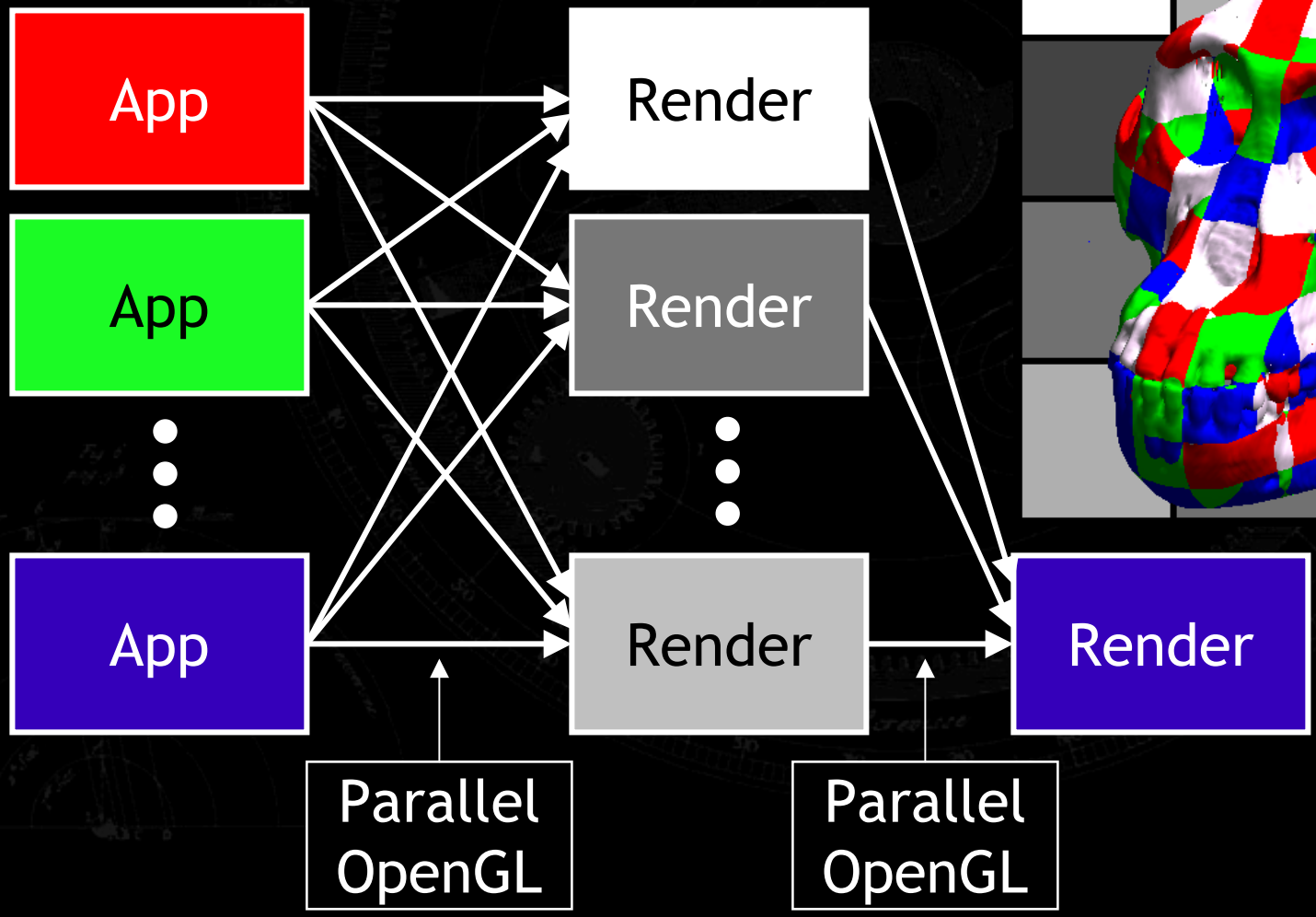
WireGL "Visualization Server"

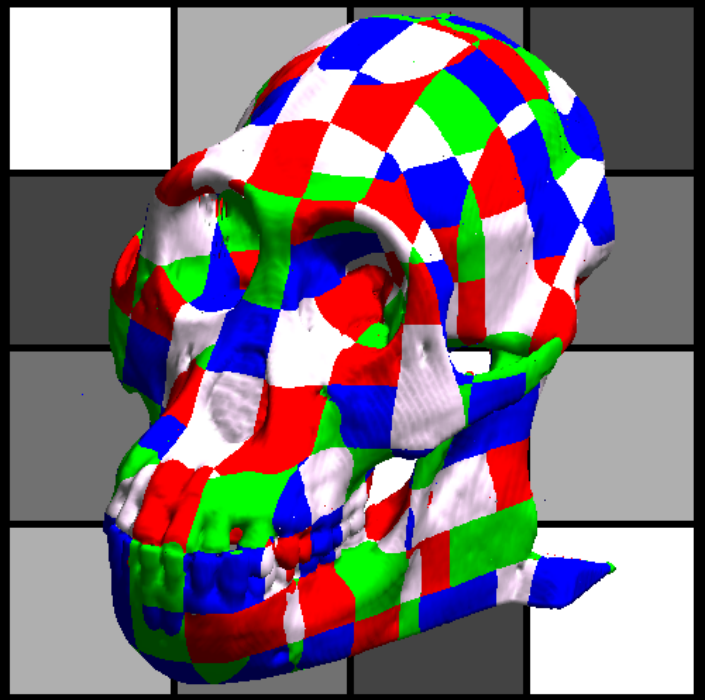




WireGL "Visualization Server"

Composite == Render!



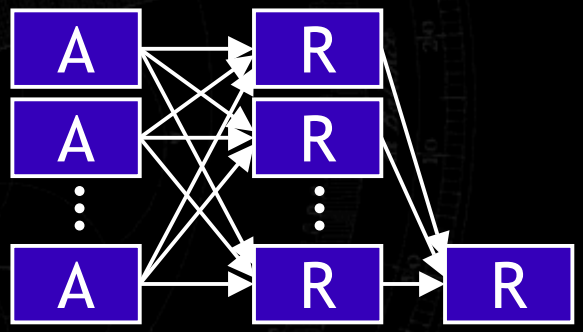


Beyond Sort-First

Where does this approach fall short?

Sort-first

- Can be difficult to load-balance
- Screen-space parallelism limited



Extensibility

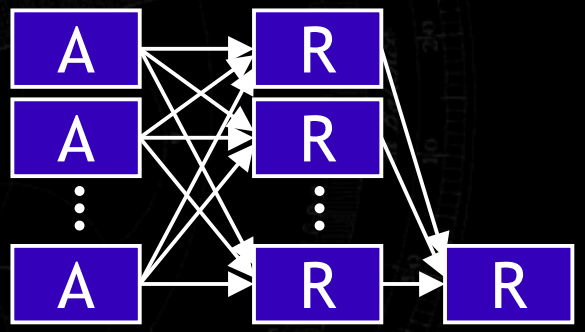
- Application bolted directly to tile/sort logic
- Sort-first paradigm inherent throughout code

We need something more flexible

Cluster Graphics as Stream Processing

The Visualization Server forms a DAG of nodes

Each node generates and/or accepts an OpenGL stream



Two available stream operations

- Tile/sort geometry, images, and state
- Render stream, possibly generating imagery stream

Other stream operations?

Other graph topologies?

Chromium

Allow arbitrary DAG's of cluster nodes

Each node may generate, absorb, or modify a stream of (extensible) OpenGL commands

Nodes are classified into two groups:

- Clients (stream sources)
- Servers (stream transformations or sinks)

Servers can be clients of other servers

Client nodes can be unmodified applications

Server Nodes

Chromium servers are similar to WireGL's

- Accept multiple incoming streams
- Resolve parallel ordering dependencies
- Dispatch stream to decoding library

Difference lies in the decoding library

- WireGL's decoder always renders stream
- Chromium allows arbitrary processing of commands
- Not just "render and display"

Client Nodes

Support existing serial applications

- Trick client into loading Chromium OpenGL DLL
- Treat OpenGL calls like any other stream
- Same arbitrary command handling as the server

Enable parallel applications

- Usually written for Chromium (no user interface)
- OpenGL API available to each node
- Parallel extensions for ordering constraints
- Each assigned a unique ID for OOB communication

Stream Processing

Each node transforms OpenGL streams

Done by "Stream Processing Units" (SPU's)

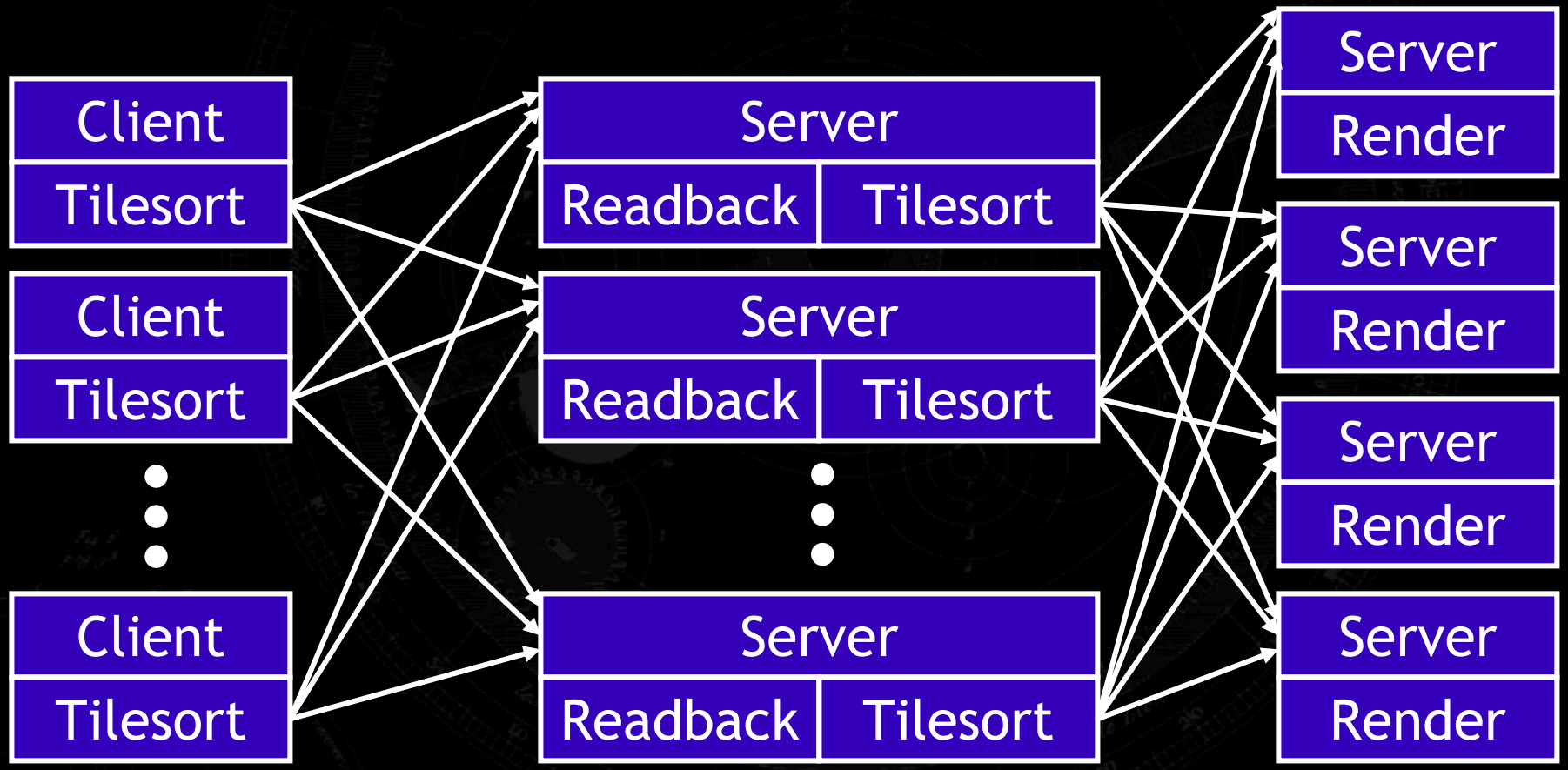
Each SPU is a shared library

- Exports a (partial) OpenGL interface
- Usually just dispatch to other SPU's

Each node loads a *chain* of SPU's at run time

SPU's are generic and interchangeable

Example: Parallel Sort-First + Bertha



SPU Inheritance

The Readback and Render SPU's are related

- Readback renders everything except SwapBuffers

Readback *inherits* from the Render SPU

- Override parent's implementation of SwapBuffers
- All OpenGL calls considered "virtual"

Other useful SPU's to inherit from:

- Error (all SPU's inherit from Error implicitly)
- Pass-through
- NOP

Example: Readback's SwapBuffers

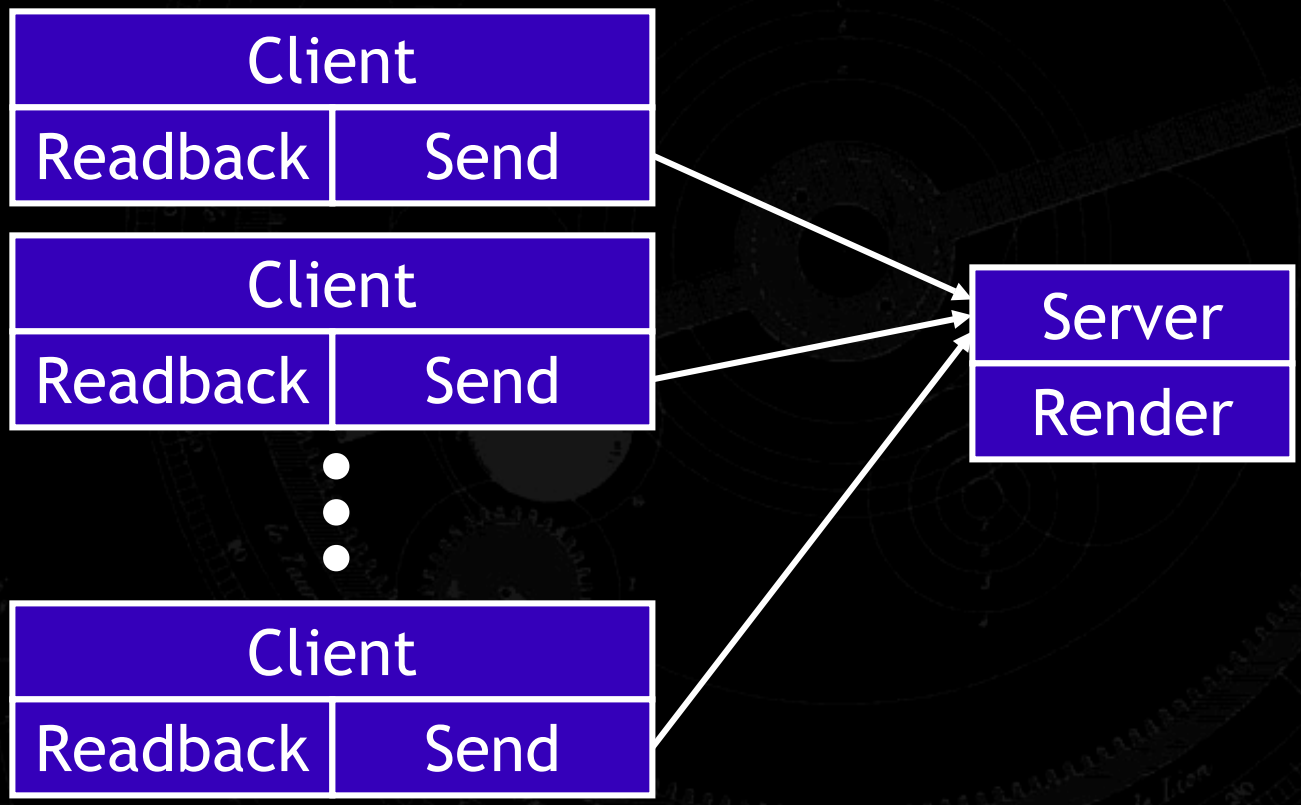
```
void RB_SwapBuffers(void)
{
  self.ReadPixels( 0, 0, w, h, ... );
  if (self.id == 0)
    child.Clear( GL_COLOR_BUFFER_BIT );
  child.BarrierExec( READBACK_BARRIER );
  child.DrawPixels( w, h, ... );
  child.BarrierExec( READBACK_BARRIER );
  if (self.id == 0)
    child.SwapBuffers( );
}
```

Note use of barriers

Easily extended to include depth composite

All other functions inherited from Render SPU

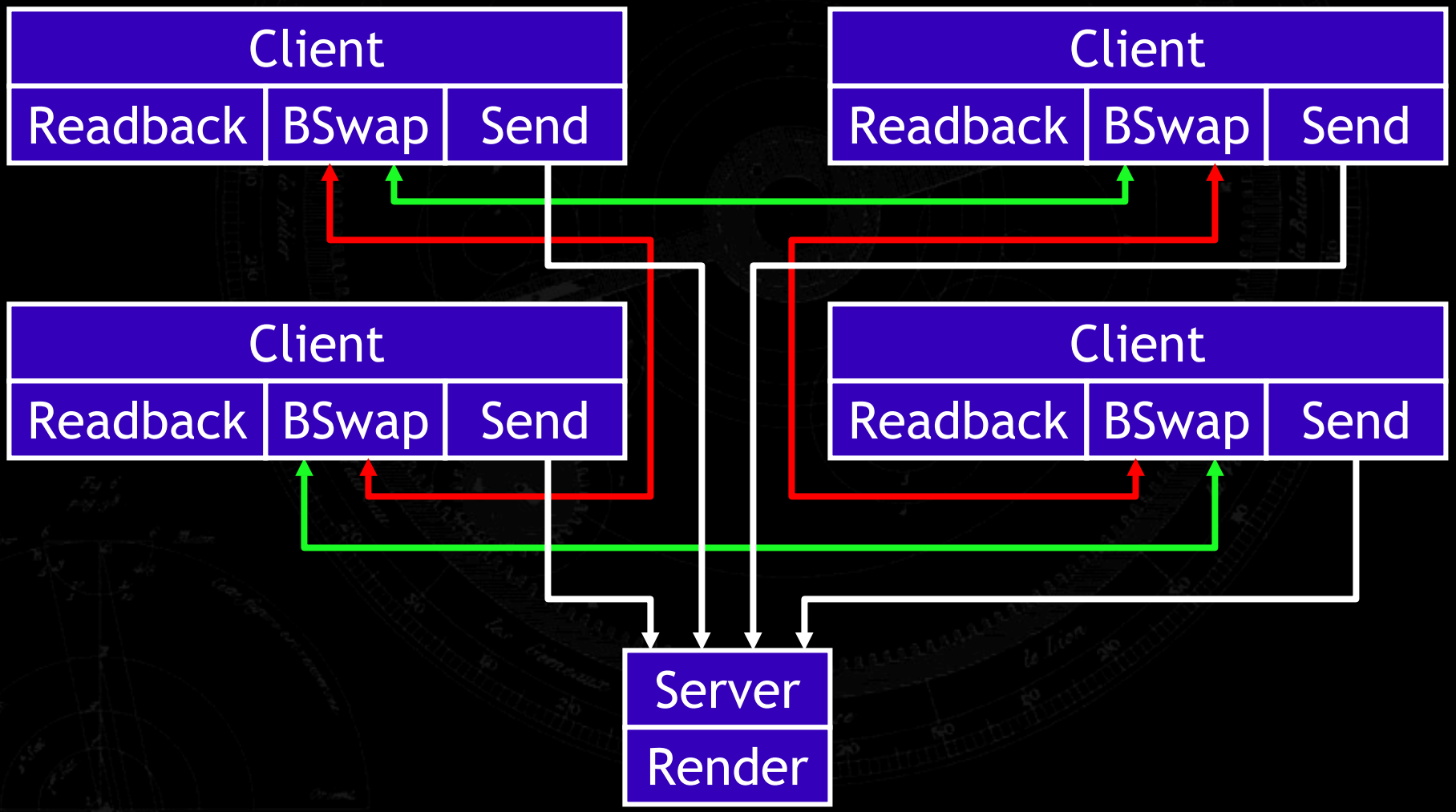
Example: Sort-Last Full



Client runs directly on graphics hardware

Readback extracts color and depth

Example: Sort-Last Binary Swap



Provided Libraries

Network

- Connection-based abstraction, easily ported

Packing

- Create WireGL-protocol network buffers

Unpacking

- Dispatch network buffers to a SPU
- Enables multipass SPU's (geometry compression)

State Tracking

- Complete OpenGL state
- Very fast incremental differences between contexts

Provided SPU's

Utility SPU's

- Error, NOP, Pass-through
- Useful to inherit from

Display SPU's

- Render, Readback (with depth)

Transmission

- Send-only, Tilesort
- Always at the end of a chain

Specialized

- Binary-swap compositor

Tee

- Useful for "tapping" streams to disk for playback/debugging

Using Chromium

Build graphics supercomputer from a cluster

- Use provided SPU's
- Use described graph topologies
- Chromium-enabled VTK, MeshTV, PV3 libraries

Experiment with new graph layouts

- Combine existing SPU's in new ways

Port to other networks

Write custom SPU's

- Wire protocol extensible
- `glHint()` convention for simple communication

Configuring Chromium

All configuration managed centrally

Configuration requests made over network

Multi-level

- Global (per-cluster)
- Per-SPU type
- Per-node
- Per-SPU instance

SPU's can discover graph topology

Configuration is scriptable

- Easy scalability experiments

Status and Timeline

Infrastructure implemented and tested

- Packing, unpacking, state tracking, networking
- Support for TCP/IP, Myrinet, file-log "network"
- Win32, Win64, Linux, IRIX, AIX (others coming…)

Configuration "mothership" working

Some SPU's written and working

- Utilities, Send-only, Render

Alpha release in early Summer (July)

Full release in early Fall (September/October)

Getting Chromium

Project housed on SourceForge

- http://www.sourceforge.net/projects/chromium
- Mailing lists for announcements, development, users

Open-source release (GPL license)

Summary

Complete framework for cluster rendering

Sort-first/sort-last systems out of the box

Testbed for new cluster rendering algorithms

- New communication topologies
- New graphics stream processing

Enable rapid integration of multiple research efforts

- Bring research results to applications more quickly
- Make research collaboration easier